FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Improving markdown decisions in non-food retail - a retailer's case study

António Cunha Seco Fernandes de Almeida

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Ana Paula Rocha, PhD

Co-Supervisor: Daniel Castro Silva, PhD

July 29, 2020

Improving markdown decisions in non-food retail - a retailer's case study

António Cunha Seco Fernandes de Almeida

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Rui Chamacho, PhD
External Examiner: Alberto Fernández, PhD
Supervisor: Ana Paula Rocha, PhD
Co-Supervisor: Daniel Castro Silva, PhD

July 29, 2020

Abstract

Products typically follow a known life cycle composed of four stages: (i) introduction; (ii) growth; (iii) maturity; (iv) decline. However, in the complex environment of retail sales, defined by fierce competition and increased customer expectations due to recent trends such as e-commerce, this traditional life cycle is rarely followed. Instead, products have short and erratic life cycles, which many times lead to unsold inventory. To increase demand, retailers typically apply markdowns, i.e., permanent price reductions. This dissertation arises in the context of a project with a large Portuguese retailer and has the primary goal of improving their price markdown strategy, with an emphasis on products reaching their end of life on three categories of their Non-Food division.

When deciding on what markdowns to set, it is crucial to know how a price change will affect demand, a relationship known as Price Elasticity of Demand (PED). PED is accounted for when defining Markdown Optimization (MDO) problems, which typically involve a sales forecasting solution and a price optimization formulation. However, the approaches are usually based on parametric modulations of demand, which underperform when faced with short life cycles and new articles with little historical data. Therefore, we proposed a threefold solution.

We started by developing a weekly sales forecasting model focused on predicting the late stages of the life cycle, to identify underperforming items. We included multiple predictors based on factors that influence demand, e.g., seasonality. We experimented with tree-based supervised learning algorithms, given their whitebox nature. Furthermore, we applied k-means clustering to group products with similar sales profiles, using each centroid as an additional predictor. We assigned new items to a predefined cluster based on their characteristics, to ensure stable predictions from the start.

We then explored the impact price had on sales, realizing markdowns affect demand directly after they are set. Based on this, we developed a PED estimation model by labeling weeks where markdowns occurred with the respective sales variation, and used supervised learning techniques to predict the outcome. As predictors, we included mostly price-related information, e.g., the price of similar products. Given the smaller dataset we also introduced AutoML, which outperformed the tree-based regressors in exchange for explainability.

Furthermore, we combined both models to formulate an MDO solution. By adding to the sales predictions the expected sales increase in response to a markdown, we computed scenarios for typical discounts (e.g., 10%, 25%), suggesting the optimal one, i.e., the minimum that ensures stock sold reaches a predefined target. The use cases supported by the predictive models are integrated within a Decision Support System (DSS) developed in parallel to this work. To ensure weekly updates , we developed a data pipeline that runs all required deployment steps.

To validate our solution, we first compared the predictive models with the approaches currently in place. We yielded an average improvement of 14.25pp in weekly sales forecasting error (measured in MAPE), and a 3.77pp improvement in stock sold estimation error. In the latter, by combining both models, we yielded an additional 2.65pp during discount periods. Additionally, we ran a pilot test alongside retail managers and evaluated the satisfaction rate of the use cases developed. The solution was generally accepted and is being extended to additional categories.

Keywords: sales forecasting, machine learning, markdown optimization, decision support system

Resumo

Os produtos seguem normalmente um ciclo de vida conhecido, composto por quatro fases: (i) introdução; (ii) crescimento; (iii) maturidade; (iv) declínio.

No entanto, no ambiente complexo do retalho, caraterizado pela concorrência feroz e pelo aumento das expectativas dos clientes devido a tendências recentes como o comércio electrónico, este ciclo de vida tradicional raramente é seguido. Em vez disso, os produtos têm ciclos de vida curtos e erráticos, o que muitas vezes leva a inventários não vendidos. Para aumentar a procura, os retalhistas normalmente aplicam markdowns, ou seja, reduções de preços permanentes. Esta dissertação surge no contexto de um projecto com um grande retalhista português, e tem como principal objectivo melhorar a sua estratégia de markdowns de preços, com ênfase nos produtos que atingem o seu fim de vida em três categorias da sua divisão Non-Food.

Ao decidir sobre que preços a estabelecer, é crucial saber como uma alteração de preço irá afectar a procura, uma relação conhecida como Elasticidade de Preço-Procura (PED). A PED é contabilizada ao definir problemas de Markdown Optimization (MDO), que normalmente envolvem uma solução de previsão de vendas e uma formulação de optimização de preço. Contudo, as abordagens são geralmente baseadas em modulações paramétricas da procura, que têm um desempenho inferior quando confrontadas com ciclos de vida curtos e novos artigos com poucos dados históricos. Por conseguinte, propusemos uma solução com três componentes.

Começámos por desenvolver um modelo semanal de previsão de vendas centrado na previsão das fases tardias do ciclo de vida, para identificar artigos com baixo desempenho. Incluímos múltiplos preditores baseados em factores que influenciam a procura, por exemplo, a sazonalidade. Fizemos experiências com algoritmos de aprendizagem supervisionada com base em árvores, dada a sua explicabilidade. Além disso, aplicámos um clustering k-means para agrupar produtos com perfis de vendas semelhantes, utilizando cada centróide como um preditor adicional. Atribuímos novos itens a um agrupamento predefinido com base nas suas características, para assegurar previsões estáveis desde o início.

Explorámos então o impacto que o preço teve nas vendas, percebendo que as baixas de preço afectam a procura directamente após a alteração. Com base nisto, desenvolvemos um modelo de estimativa PED através da marcação de semanas em que ocorreram baixas de preço com a respectiva variação de vendas, e utilizámos técnicas de aprendizagem supervisionada para prever o resultado. Como preditores, incluímos sobretudo informação relacionada com o preço, por exemplo, o preço de produtos semelhantes. Dada a reduzida quantidade de dados, introduzimos também o AutoML, que superou os regressores baseados em árvores em troca da explicabilidade.

Além disso, combinámos ambos os modelos para formular uma solução MDO. Ao acrescentar às previsões de vendas o aumento esperado das vendas em resposta a uma redução, calculámos cenários para descontos comuns (por exemplo, 10%, 25%), sugerindo o cenário óptimo, ou seja, aquele com desconto mínimo que assegura que o stock vendido atinge um objectivo pré-definido. Os casos de utilização suportados pelos modelos preditivos estão integrados num Sistema de Apoio à Decisão (DSS) desenvolvido em paralelo com este trabalho. Para assegurar actualizações

semanais, desenvolvemos um pipeline de dados que executa todas as etapas de implementação necessárias.

Para validar a nossa solução, comparámos primeiro os modelos preditivos com as abordagens actualmente em vigor. Conseguimos uma melhoria média de 14,25pp no erro de previsão de vendas semanais (medido em MAPE), e uma melhoria de 3,77pp no erro de estimativa de vendas em stock. Neste último, ao combinar ambos os modelos, conseguimos uma redução adicional de 2,65pp no erro durante os períodos de desconto. Além disso, realizámos um teste piloto juntamente com gestores retalhistas e avaliámos a taxa de satisfação dos casos de utilização desenvolvidos. A solução foi geralmente aceite e está a ser alargada a categorias adicionais.

Keywords: previsão de vendas, *machine learning*, *markdown optimisation*, sistema de apoio à decisão

Acknowledgements

The last five years have been a tremendous adventure. I could not have gone through them without the company of an exceptional set of people, to whom I sincerely thank.

To my supervisors, Ana Paula Rocha and Daniel Castro Silva, for accepting to counsel me through this final chapter and for the thorough review of this work.

To everyone at LTPLabs, in special to Bruno, for the friendship, guidance, and especially the patience: a few months ago I thought markdown was a formatting-language and that forests had actual trees.

To everyone who ever shared a handball field with me over the past decade: the names are too many to point out, but know that you have kept me sane.

To my friends from FEUP, in particular to Damas and Diogo: you were the ones calling when I fell asleep for the third time that week. Um sargento nunca abandona os seus soldados.

To my grandfather, Paulo: I envy your spirit and youth. May I one day live to write as passionately as you do.

To my family, namely my brothers Nuno and Guilherme: it has been a pleasure seeing you both grow into exceptional young adults. I may be the oldest, but not the wisest.

To Maria: for the companionship during these turbulent times, for keeping a smile on my face with your bursts of randomness, and for somehow pushing me down the stretch of this journey.

Obrigado,

António

"Quem não arrisca não arrisca, é verdade
Mas não há melhor petisco do que a liberdade"

Sanha Allen Pires *aka* Allen Halloween

Contents

List of Figures

List of Tables

List of Listings

Abbreviations

ARIMA	AutoRegressive Integrated Moving Average
ANN	Artificial Neural Networks
API	Application Programming Interface
CDSW	Cloudera Data Science Workbench
COVID-19	Corona Virus Disease 2019
CRISP-DM	Cross Industry Standard Process for Data Mining
DSS	Decision Support System
ETL	Extract, Transform, Load
GBM	Gradient Boosting Machines
HDFS	Hadoop Distributed File System
MA	Moving Average
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MDO	Markdown Optimization
ML	Machine Learning
MSE	Mean Square Error
OLTP	Online Transaction Processes
PED	Price Elasticity of Demand
PLC	Product Life Cycle
RF	Random Forests
RMSE	Root Mean Square Error
SQL	Structured Query Language
SKU	Stock Keeping Unit
XED	Cross-price Elasticity of Demand
XRT	Extremely Randomized Trees

Chapter 1

Introduction

This chapter introduces a general overview of the retail sales environment and important concepts, namely Product Life Cycle (PLC) management and Markdown Optimization (MDO), as well as the problem we propose to solve. Section 1.1 explains the context in which this dissertation arises. Section 1.2 describes the primary goals of our work, namely its research questions and research methodology. Finally, Section 1.3 describes how the rest of this document is structured.

1.1 Context

This work arises in the context of an ongoing project with a large Portuguese retailer. In total, the retailer has hundreds of stores of multiple sizes covering the entire Portuguese territory and has a business volume in the range of billions of Euros. A scale of this magnitude means their amount of products is in the thousands, which put pressure on internal logistics. Product life cycle (PLC) describes the typical pattern that products follow during their life, which is traditionally defined as having four stages: (i) introduction; (ii) growth; (iii) maturity; (iv) decline [Polli and Cook, 1969]. Identifying the stage at which a product is in is crucial information when making operational decisions, including pricing. However, in the complex environment of retail sales, defined by the evergrowing competitiveness and increased customer expectations thanks to recent trends such as e-commerce, the traditional life cycle is rarely followed. Instead, products have short and erratic life cycles, making operational decisions a hard task [Thomassey, 2014].

Despite the unpredictable setting, purchases from suppliers are typically done with months in advance, which leads to less-than-expected sales and unsold inventory. In large retailers, this can mean losses in the order of dozens of millions of Euros [Smith and Achabal, 1998]. Retailers try to reduce these losses via price markdowns. Markdowns are *"permanent price reductions in the*

price of an item in order to stimulate sales" [Dunne et al., 2013]. When establishing them, the main issues retailers face are deciding on when a markdown should be set – do it too son and revenue is lost; do it too late and sales targets may not be met – and by how much to discount – what is the minimum price change that will trigger the required demand.

The retailer found that their traditional approach to PLC management is no longer apt to its scale. There is no standardized process to monitor the performance or stock of items reaching their end of life. Consequently, pricing decisions are made in a *ad-hoc* manner based on judgemental evaluations with no analytical foundation.

When deciding on price markdowns, typical questions faced by retailers are: what is the impact of a price change in this product? What is the best price change that maximizes revenue while meeting business goals and clearance dates? These questions are aggravated by the variability of the products in terms of shelf-life, sales volume, and historical data.

Markdown Optimization (MDO) is a pricing strategy that looks to compute optimal prices for items reaching their end of life, focused on maximizing revenue while minimizing left-over stock [Coşgun et al., 2013]. Over the past few decades, computational tools have proven to be successful in helping retailers perform MDO [Chen and Simchi-Levi, 2012]. However, these solutions are mostly based on demand formulations via simple linear models that are not apt to handle the recent trends in PLC and a large amount of newly introduced products, for which there is no historical data. Moreover, these tools are not flexible to a given retailer's nuances.

Considering this, the retailer employed LTPLabs, an analytics consultancy company, to design and implement a project focused on improving the retailer's processes regarding PLC management, with an emphasis on the tail end of the life cycle, i.e., maturity and decline, where price markdowns are typically applied. In specific, they look to apply this to their Non-Food division, starting with three product categories comprised of home appliances, stationery products, and fashion apparel. This dissertation focuses on the project's section that tackles the retailer's price markdown strategy.

1.2 Goals

This dissertation's main objective is to improve the retailer's price markdown strategy through computerized assistance.

We look to achieve this by developing a set of predictive models that support the process of making pricing decisions. In particular, we look to introduce (i) a sales forecasting methodology capable of handling the challenges raised by short life cycles and newly introduced items, so that products in need of a price markdown can be identified; (ii) a methodology to estimate the sales impact of a given price markdown on a product (e.g., 10%, 25%), so that the optimal price markdown can be computed and suggested;

The solution will be integrated within a Decision Support System (DSS) that is being developed in parallel to this work. To ensure this, we also focus on developing the pipeline responsible for automatically executing the deployment steps of all use cases, from data extraction to final

model integration. With the development of this solution, we expect to enable the retailers with the necessary tools to perform informed pricing decisions with additional quality, and more efficiently.

1.2.1 Research Questions

Taking into account the goals of this dissertation the main research questions are:

- **RQ1:** *Can the retailer's approach at sales forecasting be improved through an approach based on product life cycle pattern detection?*
- **RQ2:** *Does updating a sales forecast through a price elasticity of demand estimation model provide an accurate simulation of arbitrary price markdown scenarios?*
- **RQ3:** *Does providing the retailer with the set of underperforming products and price markdown scenarios increase pricing quality and efficiency?*

1.2.2 Research Methodology

The primary purpose of this work is not only to improve the retailer's financial results within clearance sales but also to improve their efficiency when taking price markdown decisions. The developed solution must be scientifically sound but also have practical usage.

Considering this, we first perform a literature review on areas relevant to comprehend the nuances of the retail sales environment, namely Retail Sales Forecasting, methodologies to estimate Price Elasticity of Demand, and Markdown Optimization. We also cover the main topics related to Data Science because we are certain the task at hand will rely heavily on our capability to work with large amounts of data.

We measure the success of this work in two ways. First, using historical data, we benchmark the predictive power of our models against the current approach used within the retailer.

Second, we quantify the acceptance rate of the solution within the end users of the DSS, i.e., the managers of the target categories. We conduct a pilot test during which a prototype version is deployed, with the primary goal of obtaining feedback on the developed use cases. We also evaluate the answers to a survey deployed at the end of the pilot test.

1.3 Document Structure

The outline of the rest of the document is as follows: Chapter 2 aims to introduce the relevant subjects for this work, which we divide into four parts: an overview of sales forecasting in the context of retail; an in-depth look at PED and case studies that involved estimating this measure; an overview of Markdown Optimization and relevant work; an overview of Data Science concepts. In Chapter 3, we focus on the problem at hand in this dissertation, starting with an overview of the target retailer. We then present the requirements conditioning this work, followed by a description

of the solution and the tools used to achieve it. Chapter 4 takes a more in-depth look at the solution and its implementation steps. Chapter 5 presents the results obtained with this work. Chapter 6 summarizes the work developed, the main contributions and future improvements.

Chapter 2

Literature Review

This chapter extensively describes the concepts needed to follow this dissertation. Section 2.1 and respective subsections give an overview of the area of Retail Sales Forecasting, including an introduction to Time Series Forecasting, the factors known to influence demand, and typical forecasting aggregations. Section 2.2 describes the concept of Price Elasticity of Demand (PED) and derivatives, and reviews work which involved estimating PED in the context of retail. Section 2.3 introduces Markdown Optimization (MDO) and its role in supporting clearance sales, a general MDO formulation based on standard assumptions, and a review of relevant work. Section 2.4 introduces the field of Data Science, namely a general methodology to follow in data-driven tasks and a review of relevant Machine Learning (ML) concepts. Finally, Section 2.5 presents a summary of the reviewed work.

2.1 Retail Sales Forecasting

Retail markets have evolved significantly since their ancient time origins, and so did the role of retailers. From a traditional standpoint, retailers satisfy demand identified through a supply chain, functioning as intermediaries between suppliers and costumers [Niemeier et al., 2013].

However, the digital era has threatened this intermediary role. The new technologies that emerged in the late 1990s and have now matured are game-changing. This has been noticeable by the declining significance of the traditional brick-and-mortar stores and the rise of numerous retailers who take advantage of the now ubiquitous computing power as a means to understanding demand and fulfilling it [Niemeier et al., 2013].

The rise of e-commerce has simultaneously paved the way and put pressure on managers of leading retailers to invest in international expansion. Contrary to initial expectations, the complexity of cross-border developments proved hard to fulfill and lead to relatively unsuccessful attempts, such as Tesco's endeavor into France [Palmer, 2004][1].

In this light, many retailers focused instead on improving their internal efficiency and differentiation to increase their value offer. With the help of technology, retailers streamlined their operations and opened more efficient stores. Such was the case of Walmart, which according to Johnson and Bradford had in 1987 "*a market share of just 9 percent but was 40 percent more productive than its competitors as measured by real sales per employee*" [Johnson and Bradford, 2002]. This is also the basis for the strategy currently in place at the retailer in this work.

The large retailers' industry is ever-evolving in both structure and in the challenges it faces. The high scale, low margin nature of this competitive market makes it inherently challenging. The past decade has been marked by the rise of online shopping, the increase of fiercer competition, and, consequently, higher customer standards. In this scenario, efficient internal operations are vital in fulfilling customer expectations while remaining profitable [Niemeier et al., 2013].

Consequently, forecasting is a typical task in business, where it helps to inform many decisions across all company levels: number of people to hire, the scheduling of production, number of stock orders, right down to the number of office supplies that should be purchased [Chase, 2013]. It provides a guide to long term strategic planning while also ensuring daily operations [Hyndman and Athanasopoulos, 2018]. In particular, retailers are typically interested in Sales Forecasting.

2.1.1 Time Series Forecasting

Forecasting is the science of predicting future events. Many things can be forecast, but some are easier than others. The reality is that a forecast's accuracy is limited by the nature of what we are trying to forecast [Gilliland et al., 2016]. Hyndman and Athanasopoulos argue an event's predictability depends on multiple factors, including [Hyndman and Athanasopoulos, 2018]: (i) how much we understand the factors that contribute to it; (ii) how much data is available; (iii) whether the forecasts can affect the thing we are trying to forecast.

Forecasting methods range from really simple, e.g., using the most recent observation as a prediction, to highly sophisticated approaches, e.g., econometric systems and neural networks [Fildes et al., 2019]. The appropriate methods to employ depend mainly on the problem at hand and the available data. When there is no data or it is not relevant to the forecasts, then *Qualitative Forecasting* methods must be used, such as integrating expert knowledge, or market research. If historical information *is* available, then *Quantitative Forecasting* methods come into play. These methods assume it is reasonable to expect that aspects of past events will continue to happen [Hyndman and Athanasopoulos, 2018].

Most quantitative prediction problems deal with data that is collected regularly over time (e.g., hourly, daily, weekly), typically mentioned as *Time Series* data. Some examples of time series data

[1] Tesco has later successfully expanded into multiple international markets.

include: a person's heart rate, a company's stock price, and weekly rainfall data [Senter, A., 2008]. When forecasting time series data, the objective is to predict how the sequence of observations will continue. As an example, Figure 2.1 shows the total world consumption of copper from 1951 to 1968, where the yellow background show forecasts for the next seven years. It is noticeable how the model captured the increasing trend of the historical data and replicated it for the next seven years.

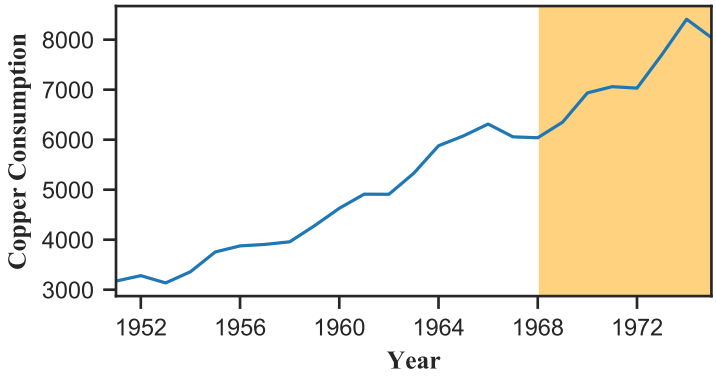


Figure 2.1: The total yearly consumption of copper, in thousands of metric tons, from 1951 to 1968, with a forecast ranging from 1968 to 1975 (yellow background). Data from [Gill, 2000].

Predictor Variables

The simplest forecasting methods take into account just the values of the variable we're trying to predict, e.g., copper consumption. This approach yields good results at extrapolating trend and seasonal patterns, but ignores other factors that might affect the target variable. To overcome this, *predictor variables* are often used in forecasting tasks. For example, we could argue that copper consumption (CC) depends on its price, the price of competing materials (e.g., aluminum), and the overall economy. In other words, the forecasting model would be a function of the predictor variables plus an "error" term, which accounts for random variations and other variables not considered (see Equation 2.1):

$$CC = f(\text{"price of copper"}, \text{"price of aluminum"}, \text{"economy's strength"}, \text{"error"}) \tag{2.1}$$

These models are typically called causal or explanatory models because they attempt to explain the variation in the target variable instead of just predicting it. Their usefulness comes from incorporating information about relevant variables other than just the target one, while estimating their relationships. However, they are not a silver bullet: in many cases, we do not understand the system we are trying to predict, and the relationships between the variables can be incredibly complicated [Hyndman and Athanasopoulos, 2018].

To sum up, the model used in a forecasting scenario depends mostly on the problem at hand, the data available, and the end-use of the forecasting model. In Section 2.4, we take an in-depth look at a generic framework to tackle data-based tasks.

2.1.2 Sales Forecasting Aggregation Levels

Mentzer and Moon define Sales Forecasting as the *"projection into the future of expected demand, given a stated set of environment conditions"* [Mentzer and Moon, 2005]. As the definition suggests, Sales Forecasting is used interchangeably with Demand Forecasting. In practice, the goal of Sales Forecasting is to project *demand*, that is, to predict what the customers want, so the company can better prepare to achieve sales near that level.

Retailers rely on Sales Forecasting for multiple sections of future planning. At the operational level, they are input for stock purchasing and replenishment decisions, as well as essential factors in marketing, accounting, and production planning. At the administrative level, they influence the "big picture" strategy that a company follows [Fildes et al., 2019].

Each sector also has different forecasting scenarios. For instance, at the operational level, forecasts ensure daily operations, and are therefore designed and presented at lower aggregation levels, e.g., "weekly sales for product X at store Y". At the administrative level, forecasts are input for company-wide decisions, so they are observed at high aggregation levels, e.g., "monthly sales for stores in a region."

The needs of the end-users must be taken into account when designing forecasting models. In the context of retail, common use cases and respective aggregation levels include:

- **Market-level:** regards the total sales of a retail category, market, or entire industry, in a given region or country. Usually employed by large retailers, market-level forecasts aid in evaluating market shifts and trends over extended periods, i.e., months, quarters, years, assessing the impact these may have on their own sales [Alon et al., 2001]. Strategic development decisions rely on these forecasts, e.g., "what is our 5-year plan?", as they bring insights into the potential of certain regions or business models. Market-level forecasts are characterized by strong trend and seasonal patterns, and are traditionally modeled with time series approaches, such as exponential smoothing and AutoRegressive Integrated Moving Average (ARIMA) [Fildes et al., 2019]. These have proven to be efficient when macroeconomic conditions are stable. Under volatile conditions, some researchers claim nonlinear models, e.g., Artificial Neural Networks (ANNs), perform better [Alon et al., 2001].

- **Chain-level:** focuses on the sales of a given market chain. These forecasts are usually employed one year ahead, and serve as inputs for company financial management decisions. Generally, the employed models are similar to those used in market-level forecasting.

- **Store-level**: large retailers usually have numerous stores of different formats, spread along various regions (or countries). Naturally, store sales are influenced by multiple factors, including location, local economy, the presence of competition, and local events. This raises

the need for sales forecasts at the store level, which are employed to support distribution and workforce planning, financial control, among others. Alternatively, retailers forecast sales of new potential locations while analyzing where to open new stores. Typical approaches consist of time series and regression-based models.

- **Product-level**: predicting demand for each item across a given market chain, and many times, for each store. In practice, this means projecting sales for a large amount of products over a short forecast horizon, e.g., daily or weekly. For a large retail chain, these are huge numbers, e.g., Walmart deals with over one billion Product x Store combinations [Seaman, 2018]. Because of this, some retailers prefer to forecast directly at a higher level of their product hierarchy (this is typical of fashion retailers) [Caro and Gallien, 2012]. Nevertheless, these forecasts are essential for many operational decisions, e.g., space allocation, pricing, and stock replenishment. Due to this dependence, forecasting error at this level directly influences the retailer's service: overestimated sales lead to high inventory costs, while underestimated sales lead to stock-outs (and lost sales).

2.1.3 Factors that Influence Demand

Product demand is strongly impacted by numerous factors, which make the overall sales very volatile and difficult to predict. These factors are challenging to identify, and their impact is hard to measure [Thomassey, 2014]. However, they are essential in predicting and understanding demand, and the most relevant ones are integrated into forecasting models. Factors commonly known to drive demand include [Fildes et al., 2019, Thomassey, 2014]:

- **Macro-Economic Data:** the performance of the retail industry is influenced by the generic economic environment of its country or region. Macroeconomic factors such as per capita income, unemployment rate, and inflation rates influence the demand for the retail industry [Zakaria, 2018].

- **Seasonality:** retail sales show strong seasonality. It is common for products to show seasonal cycles of different lengths, e.g., annual and weekly. For instance, ice cream sales are high during summer (and low during winter), and tend to sell more during the weekend [MacDonald, 2003].

- **Weather:** some retail products are strongly affected by temperature or other weather conditions. For instance, soft drinks are known to be in higher demand when the weather is hot [Dubé, 2004]. The relationship can be non-linear: the demand for soft drinks increases when temperature raises, but might decrease when reaching extreme heat (customers switch to water). Additionally, the weather itself needs to be forecasted, reducing the usability of its predictive power, unless for (very) short-range forecasts.

- **Price:** the price at which a product is sold directly influences its demand, i.e., Price Elasticity of Demand (PED), and possibly the demand for similar or complementary products, i.e.,

Cross-Price Elasticity of Demand (XED). As a natural driver for demand, price is usually incorporated into forecasting models. However, the effects of the large variety of promotion types that retailers employ, e.g., "buy one get one free" and "50% off", are hard to capture and model explicitly. We take an in-depth look at PED and related concepts in Section 2.2.

- **Calendar Events:** proximity to holidays, festivals, and special activities (e.g., sports matches) strongly affect retail sales. Events that occur at regular intervals, e.g., Christmas, can be interpreted (and therefore modeled) as seasonality, unlike ones that shift in the calendar year, e.g., Easter, music festivals.

- **Product Characteristics:** the characteristics of a product and the way it presents itself to customers are also relevant. In the context of perishable products, e.g., fruit and vegetables, key characteristics include freshness and physical conditions, which are hard to translate into a forecasting model [Barbosa, 2018]. This is particularly relevant for apparel items: color, design, their match with current trends, among others, are critical factors for demand [Thomassey, 2014].

- **Product Stock:** particularly in the fashion industry, it has been documented that the demand rate for an article usually declines when the inventory goes bellow a certain level – a phenomenon known as *Broken Assortment Effect* [Caro and Gallien, 2012]. This happens due to the availability of a generic product but unavailability of certain colors or sizes, which is likely to occur late in the season as inventory level declines [Khouja, 2016]. Furthermore, higher inventory levels might allow more attractive displays and thus increase sales, in addition to avoiding stock-outs [Boada-Collado and Martínez-de Albéniz, 2020].

2.2 Price Elasticity of Demand

A retailer's success is driven by how well it can fulfill its market's demand. Introduced by Marshall, the Law of Demand states that as the price of a good increases, quantity demanded decreases [Marshall, 1898]. Symmetrically, as the price decreases, demand is expected to increase. In the context of retail, this puts pricing decisions among the crucial ones.

Accordingly, in order to make informed decisions regarding price, it is essential to measure the relationship between price and demand. This is achieved by calculating Price Elasticity of Demand (PED). Initially defined by Edgeworth and Marshall, it describes how changing the price of a product affects the demand for itself - *own price elasticity of demand*, or other products - *cross-price elasticity of demand* [Edgeworth and Marshall, 1891].

ΔQ_i	Change in quantity demanded for product i
$\overline{Q_i}$	Average quantity demanded for product i
ΔP_i	Price change for product i
$\overline{P_i}$	Average price of product i

2.2.1 (Own) Price Elasticity of Demand

Own Price elasticity of Demand (PED) is calculated by comparing the percentage change in a product's demand facing a change in the product's price (Equations 2.2 and 2.3):

$$\text{"Own price elasticity of demand"} = \frac{\text{"Percentage change in quantity demanded"}}{\text{"Percentage change in price"}} \tag{2.2}$$

$$PED_i = \varepsilon_i = \frac{\%\Delta Q_i}{\%\Delta P_i} = \frac{\Delta Q_i / \overline{Q_i}}{\Delta P_i / \overline{P_i}} \tag{2.3}$$

Because a positive change in price leads to a negative change in quantity, price elasticity of demand is almost always negative [Parkin et al., 2014]. For this reason, it is typical to look only at its absolute value. Steenburgh et al. present an intuitive interpretation of its value: "*the higher the absolute value of the number, the more sensitive customers are to price changes*" [Steenburgh and Avery, 2010].

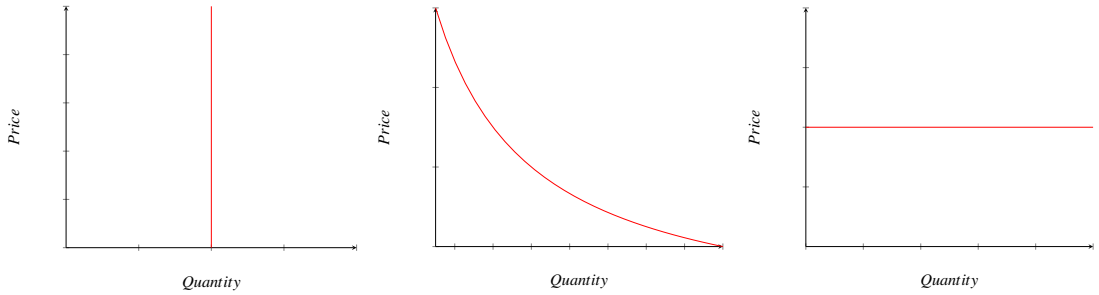
Figure 2.2 covers the edge cases of PED. Few products apply to these, but they are the reference point for the most common, and therefore, relevant cases. In Figure 2.2a, the demand does not change regardless of the price. In this scenario, PED is zero and is known as *perfectly inelastic demand*. An example of this case is a life-saving drug that people will pay any price to obtain: even if its price were to rise substantially, the quantity demanded would not diminish.

When the demand changes proportionally to price, i.e., a percentage change in price leads to an equal percentage change in demand, PED is 1, and the product is said to have a *unit elastic demand*. Figure 2.2b is an example of such a case.

In situations between Figure 2.2a and Figure 2.2b, where the percentage change in quantity demanded is less than the percentage change in price, PED is between zero and 1. The product is said to have *relatively inelastic demand* and represent the elasticity of most goods. Known examples include housing and food.

If, for a small price change, the quantity demanded changes by an infinitely large percentage, then the price elasticity of demand is infinity. Figure 2.2c illustrates this case, and the product is said to have *perfectly elastic demand*.

In situations between Figure 2.2b and 2.2c, where the percentage change in quantity demanded is greater than the percentage change in price, price elasticity is greater than 1. The product is said to have *relatively elastic demand*, that is, it responds significantly - but not disproportionally - to price changes. Known examples include cars and furniture.



(a) Perfectly inelastic demand (b) Unit elastic demand (c) Perfectly elastic demand

Figure 2.2: Edge cases of elasticity of demand.

2.2.2 Cross-Price Elasticity of Demand

Retailers have long been aware of other factors that influence a good's demand other than its own price. For instance, changes in prices of related goods can affect the demand for a particular product [Carbaugh, 2010].

Considering this, an extension of own-price elasticity of demand is that of *cross-price elasticity of demand* (XED). It is a measure of the way the demand for a good is affected, given a change in the price of another good [Parkin et al., 2014]. Similarly to PED, it is calculated as follows, between products i and j (Equations 2.4 and 2.5):

$$\text{"Cross price elasticity of demand"} = \frac{\text{"Percentage change in quantity demanded for } i\text{"}}{\text{"Percentage change in price of } j\text{"}} \tag{2.4}$$

$$XED_{i,j} = \varepsilon_{i,j} = \frac{\%\Delta Q_i}{\%\Delta P_j} = \frac{\Delta Q_i / \overline{Q_i}}{\Delta P_j / \overline{P_j}} \tag{2.5}$$

XED can have positive or negative values. In a positive case, the products in are said to be *substitutes*. For instance, let the increase in the price of pizza by 20% lead to an increase in the demand for burgers by 10%. Following equation 2.5, cross-price elasticity of demand for burgers with respect to pizza is $\frac{10\%}{20\%} = 0.5$. Because this value is positive, pizza and burgers are considered *substitutes*.

Symmetrically, when XED is negative, the products in case are said to be *complements*. Following a similar example, let an increase of 15% in the price of soda lead to a decrease in demand for burgers by 10%. Again, following equation 2.5, XED for burgers with respect to soda is $\frac{15\%}{-10\%} = -1.5$. In this scenario, burgers and soda are *complements*.

2.2.3 Case Studies

Despite the intuitive definition, an item's PED varies depending on other determinants, e.g., necessity and availability of substitutes, making it a complicated measure [Parkin et al., 2014]. In

practice, multiple methods are used to estimate PED, mostly based on analysis of historical sales data and customer preference surveys [Png, 2002].

In general, PED estimations are used as a means to improve the pricing strategy of products in a particular industry. Remarkable use cases include electricity markets [Babar et al., 2015, Thimmapuram and Kim, 2013], hotels [Hormby et al., 2010], water [Olmstead et al., 2007], and car rentals [Pachon et al., 2006].

We are interested in applications of PED estimation models on general-purpose, fashion, and food retail. In these areas, there are some studies available, mostly focused on fashion retail, and refer to PED estimation as part of a broader goal, typically a Dynamic Pricing or Markdown Optimization (MDO) system. Markdown Optimization, or Clearance Sales Pricing, is a subset of Dynamic Pricing focused on optimizing the sales revenue of items under clearance. We take a more in-depth look at MDO in Section 2.3.

An overview of the analyzed studies is presented in Table 2.1. We cover case studies in the context of MDO because it is commonly where price elasticity of demand estimation is an explicit step of the approach.

Table 2.1: Reviewed PED case studies.

Study	**Area**	**PED Estimation Approach**
ZARA [Caro and Gallien, 2012]	Fast-fashion	Linear Regression
Zaroour et. al [Zaarour et al., 2016]	Cosmetic Retail	Non-Linear Regression
Bouzdine-Chameeva and Hill [Bouzdine-Chameeva and Hill, 2015]	Retail	Not Specified
Rue La La [Ferreira et al., 2016]	Fast-fashion	Decision Trees

Caro and Gallien present the design and implementation of a pilot test system deployed at Zara[2] with the primary goal of improving their clearance sales pricing process [Caro and Gallien, 2012]. Because of Zara's fast-fashion model, where a great variety of products are typically sold in short life cycles, almost no historical price response data are available at the start of each clearance period. The authors estimated PED based on features shared by articles sold in previous seasons, followed by an updated estimation based on real sales data. They group similar items and assume PED is constant and identical among the products in each group. The chosen demand parameters are fitted in an exponential form function, which include the age of the product, purchase quantity, previous demand, and the broken assortment effect (see Section 2.1.3). It includes PED as one of the regression coefficients, which are estimated using linear regression. However, PED is only estimated after the first week of clearance sales.

Zaarour et al. present their MDO work for out-of-season items at an anonymous U.S. based retailer with more than 6000 stores. Demand is modeled using weekly sales volume and price information via regression analysis. In particular, they experimented with multiple regression

[2] https://www.zara.com/

functions, including linear, quadratic, and a power function. The power function $Q = \alpha P^{\beta}$ yielded the best fit, hence estimating PED via regression coefficients (β) (where Q is sales, and P is price). A comparative study is performed between 6 obsolete cosmetic products, with the least and most elastic items showing increased sales volumes of 3 and 6 times higher when the price is reduced by 50%.

Bouzdine-Chameeva and Hill develop two models for retail clearance pricing based on a model introduced by Smith and Achabal [Smith and Achabal, 1998], with the main difference that it does not map demand as a function of stock amount [Bouzdine-Chameeva and Hill, 2015]. The firsts model assumes demand is deterministic, the latter, stochastic. Similarly to the Zara study, the deterministic model fits demand under a function with an exponential form: in particular, a function of price, seasonal factor, and the PED. The stochastic model adds a random multiplicative error term. The authors do not describe how price elasticities are estimated: we assume they are estimated as per the original formulation, i.e., a mixture of regression analysis and judgemental input [Smith and Achabal, 1998]. In general, results indicate that the deterministic model is too aggressive, i.e., its predictions are either too low or too high.

Ferreira et al. present their work with online retailer Rue La La on sales predictions and dynamic pricing [Ferreira et al., 2016]. Rue La La[3] is an online-only fast-fashion retailer that faces similar problems to those of the Zara case, e.g., the need to predict demand for items without historical sales. The general approach for the demand prediction model is to use sales data to predict demand for each *style*[4] sold in the past. Using these results, they predict the future demand for each corresponding "product type", i.e., a set of *styles*. The authors do not directly estimate PED but use three price-related features that implicitly integrate price changes: the price of the *style*, the percent discount off, and the relative price of competing *styles*[5]. We argue including the price of competing *styles* is an indirect representation of XED. For new *styles*, the authors build a regression model based, among others, on the mentioned price features. The regression technique that yielded the best results was decision trees with bagging, outperforming parametric regressors. The authors argue that decision trees are capable of identifying non-linear effects, e.g., when demand increases with price as a signal of quality. The final demand models and price optimization were part of a field experiment that reported an increase in revenue of 9.7% for new *styles*.

2.3 Markdown Optimization

From a modern perspective, Dynamic Pricing is a pricing strategy where the price of a good or service is flexible and changes in response to current market demand. The price changes are based on algorithms that take into account factors such as traffic, competitors' prices, and weather

[3] https://www.ruelala.com/

[4] A *style* is a reference to a set of identical items with all possible sizes.

[5] Computed as the price of the *style* divided by the average price of all competing styles.

information [Omnia Retail, 2020]. In retail, Dynamic Pricing is steadily converting into a best practice as the availability of data and specialized software grows [Chen and Simchi-Levi, 2012].

The main capability that Dynamic Pricing enables is to adapt prices according to the economic principles of demand, which are based, among other factors, on Price Elasticity of Demand. Ingenbleek et al. recognize this approach as the superior pricing strategy, which they refer to as *customer value-based pricing* [Ingenbleek et al., 2003].

Efficiently managing a retail business involves decisions on multiple fronts, such as inventory management, pricing, and resource allocation [Kumar and Patel, 2010]. Reliable forecasts are crucial in supporting these decisions. However, retailers typically buy their inventory with long lead times, on fixed amounts, and based on initial sales forecasts. When these are not accurate and there is left-over stock, retailers usually apply price markdowns as an attempt to increase demand [Gupta et al., 2006].

2.3.1 Overview

Considering this, Markdown Optimization (MDO), also known as Clearance Sales Pricing, is a subset of Dynamic Pricing with a focus on items reaching their end of life. Its goal is to maximize revenue from the target items while minimizing excess inventory. Like Dynamic Pricing, it takes into account multiple factors, such as stock levels, life cycle and season trends, and the Price Elasticity of Demand.

Figure 2.3 illustrates the time horizon over which a typical sales season happens. During the *off seasons*, a product or set of products is not active. The *peak season* is when products are being sold at their full price, and potentially with stock replenishments. The *clearance season* begins at the first markdown and lasts until the *outdate*, i.e., when the remaining inventory stops being sold [Zhao and Zheng, 2000]. That is when price markdowns come into play: the difficulty lies in setting the right trade-off. Setting the price too low means loosing potential profits (or no profit at all); setting the price too high leads to useless inventory. That is why, despite being a common technique, "*retailers hate markdowns*" [Merrick, 2001].

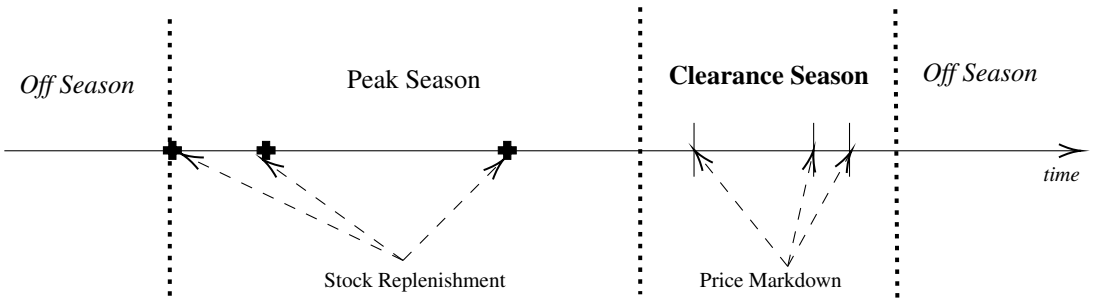


Figure 2.3: Typical season sequence for a retail product.

Smith claims the percentage of sales under markdown has been steadily increasing over the years, and now comprises one third of retail sales, identifyinh this as a result of recent trends in the

retail industry [Smith, 2015]: (i) an increase in the number of SKUs; (ii) a more significant proportion of "fashion" items; (iii) shorter selling seasons; (iv) more store brand label merchandise.

These trends increase the importance of setting optimal markdowns, but also the difficulty in doing so. Shorter selling seasons decrease the amount of historical data available to assess demand, and limit the time window to apply markdowns.

Traditional approaches to MDO involved "rule of thumb" techniques taken by managers based on domain experience [Neelakantan et al., 2007, Smith, 2015]. The rationale is simple and looks to exploit known seasonalities, e.g., the demand for Christmas trees falls rapidly near the end of the season. The markdowns are usually set the same for all the stores, regardless of inventory values. The approach leads to questionable results because it does not account for region-specific seasonalities or different stock amounts across stores.

The consistent decline in computing costs and the availability of transaction data paved the way for a computerized approach with successful results, as early as the late 1990s [Smith and Achabal, 1998]. Over the last decade, some commercial solutions have risen to tackle this problem. Recent and available examples include IBM Commerce[6], Solvoyo[7], and Oracle Retail[8]. We take a more in-depth look at IBM's approach in Section 2.3.4. However, these solutions are usually part of a broader retail management solution and are not customizable to tackle each retailer's nuances. Considering this, retailers have developed tailored strategies according to their business model and logistics requirements (e.g., [Caro and Gallien, 2012, Smith and Agrawal, 2017, Zaarour et al., 2016, Mantrala and Rao, 2001]).

More specifically, designing and implementing a Markdown Optimization system requires the interoperability of multiple components. Smith lists the high-level modules that a successful system should comprise [Smith, 2015]:

1. A sales forecasting model;
2. A Markdown Optimization algorithm specified at the item and store level;
3. A way to measure the effectiveness of the system.

2.3.2 Modeling Assumptions

Elmaghraby and Keskinocak [Elmaghraby and Keskinocak, 2003] mention that three characteristics of the market environment define the type of dynamic pricing problem at hand:

- **Replenishment vs. no replenishment of inventory (R/NR):** whether or not it is possible to add stock during the sales period changes the decision approach.
- **Dependent vs. independent demand over time (D/I):** customer's knowledge about a product can affect their decision to purchase and willingness to pay. This is true for most necessity items, e.g., milk and break, which costumers frequently buy. For seasonal or fashion

[6] https://www.ibm.com/client-tools/ibm-digital-commerce
[7] https://www.solvoyo.com/
[8] https://www.oracle.com/industries/retail/

goods, the selling horizon is too short to allow for significant knowledge acquisition that has an impact on demand;

- **Myopic vs. strategic customers (M/S)**: whether or not customers are strategic in their purchases influence the pricing decision.

Considering this, MDO problems are commonly treated in the context of no replenishment, independent demand (NR-I). In other words, there are no stock replenishments (the goal of MDO is to minimize left-over stock), and demand is independent over time (the short-selling-window argument is applied to clearance seasons). More specifically, Zhao and Zheng present common assumptions of NR-I markets [Zhao and Zheng, 2000]: (i) the retailer functions in a market with imperfect competition; (ii) stock is limited and is not replenished; (iii) buying inventory is a sunken cost; (iv) demand decreases with price; (v) unsold items have a salvage value, i.e., they can be sold by the retailer to a wholesaler. Consequently, pricing decisions are mainly driven by PED. These observations are consistent with those of Smith, who adds that demand declines at the end of clearance season because items become "*out of season*" [Smith, 2015].

2.3.3 Model Formulation

With the assumptions above, we can formulate a generic MDO problem. We define the demanded quantity Q as a generic function of unit price p, and period t. For each SKU i, the primary goal is to determine the unit price p_i to charge in each period t. Therefore, revenue R is simply multiplying the demanded quantity Q by the unit price:

$$R = Q_i * p_i$$

where our goal is to maximize R over all periods. Considering this, following the notation in Table 2.2, we define the problem as follows:

$$\begin{aligned}
\text{maximize} \quad & \sum_{t_i=t_0}^{t_e} R_i = p_{i,t}.Q(p_i,t) \\
\text{subject to} \quad & Q(P_i,T) \leq I_0, \\
& p_{i,t}^{buy} \leq p_{i,t} \qquad ,\forall t \in T, \\
& p_{i,t_j} \leq p_{i,t_i} \qquad ,\forall t_i < t_j \in T, \\
& p_{i,t} > 0 \qquad ,\forall t \in T, \\
& I_0, p_i^{buy} > 0
\end{aligned} \tag{2.6}$$

The restrictions presented are, by order of appearance: (i) the total units demanded are less than the product's inventory; (ii) products cannot be sold for less than the price the retailer paid for them; (iii) once a price goes down, it cannot go back up; (iv) item prices and initial inventory values are positive. This is an adapted version featuring the common parameters of the formulations proposed by [Zaarour et al., 2016, Smith, 2015, Vauhkonen, 2018]. It should be interpreted as

Table 2.2: Nomenclature for the generic markdown optimization problem.

	Sets
T	Set of periods
P_i	Set of prices for product i over T, i.e. $\{p_{i1}, p_{i2}, ..., p_{it}\}$
$Q_i(p)$	Estimated demand for product i at price p and period t
	Parameters
t_0	Starting period
t_e	The outdate, i.e., the final period
t	An arbitrary period, $t_0 < t_e$
I_0	Inventory amount at the starting period
p_i^{buy}	Price that the retailer bought product i for
	Decision Variables
$p_{i,t}$	Price of product i at period t

the basis for more elaborate proposals, which take into account other factors (e.g., salvage prices, inventory-dependent demand).

2.3.4 Case Studies

In the recent literature, we can find some case studies that are relevant to our context. We are interested in MDO applications that tackle retailers with multiple stores, preferably in the hundreds. An overview of the covered studies is presented in Table 2.3. In those cases, we now present them from their markdown optimization specifications.

Table 2.3: Reviewed MDO case studies.

Study	**Novelty Characteristics**
Smith (2017) [Smith and Agrawal, 2017]	Inventory dependant demand, optimize inventory allocation
Zarour et al. [Zaarour et al., 2016]	Cluster analysis, predefined price set
IBM Commerce [Harsha et al., 2019]	Cross-channel relationships, Stochastic demand

Smith and Agrawal describe their work in MDO deployed at a large retailer [Smith and Agrawal, 2017]. Besides price optimization, they tackle the redistribution of available stock among the stores. The approach incorporates three levers: what prices to set, which stores to stock, and how to stock the select stores. The procedure is three fold: (i) the optimal price is calculated for each SKU, at the store level; (ii) the optimal inventory redistribution is calculated for a set of stores, assuming the optimal prices are used; (iii) optimize the set of stores to replenish, assuming optimal price and optimal distribution. Two pricing policies are introduced: a multiple price trajectory, and a single (optimal) price, introduced on an earlier work by Smith [Smith, 2015]. Demand is formulated with exponential price sensitivity. The models are tested under simulated data with inventory dependent demand and assuming 100 stores. The results indicate that simultaneously considering these three levers lead to significant profit increases. In general, the

single constant price model proved to be just slightly worse than the continuous price trajectory. If seasonality is the same across all stores, then the optimal price is the same in every store, which is valuable to retailers, who typically prefer to have a standardized price across stores.

Zaarour et al. present their work with an undisclosed U.S. based retailer with over 6000 stores, developing two models [Zaarour et al., 2016]. The first model uses multi-period nonlinear programming to determine the optimal markdown strategy for continuous prices. The second limits the price options to a set of predetermined prices, solving it through a linear programming model. As mentioned in Section 2.2.3, demand is modeled as a function of price in exponential form, modeled through non-linear regression. Furthermore, price-sales entries with similar values are grouped via k-means clustering, replacing the original data with the respective centroid to increase data point significance. This is done individually per SKU as long historical data was available. They compare results with the retailer's current strategy and indicate revenue improvements per SKU from 8.6% to 22.6%.

Harsha et al. [Harsha et al., 2019] present the implementation of a Dynamic Pricing system within a large omnichannel retailer, *i.e.*, a mix of online and traditional brick-and-mortar shopping. The authors assess cross-channel interactions, i.e., how price markdowns in the online shop affect the sales of the conventional stores, and the opposite. They formulate a dynamic stochastic optimization model and multiple pricing policies (i.e., deterministic vs. "robust" pricing), and compare the results. The pilot test conducted at the retailer was a success, reporting an estimated 13.7% in clearance revenue increase. This led to the integration of the system as part of a commercial MDO solution by IBM as part of its e-commerce cloud platform.

2.4 A Data Science Approach

Developing analytical systems targeting retailers typically involves direct interaction with their information systems, which requires knowledge of data-centric tasks[9]. The term Data Science dates back to the late 1990s when the number of large and complex datasets available began to increase, which raised the need for statisticians to join computer scientists in rigorously analyzing these massive databases [Wu, 1997]. Since then, the tremendous growth in the amount of data generated has increased the need to exploit it. One of the approaches involves building accurate prediction models using Machine Learning techniques, which nowadays drive decision making in nearly all parts of modern societies [Kelleher and Tierney, 2018].

2.4.1 CRISP-DM

The complexity of handling the multiple stages of a data related task called for a more structured methodology. The Cross Industry Standard Process for Data Mining (CRISP-DM) is a standard approach for Data Mining projects, which is often also adopted for Data Science projects [Kelleher and Tierney, 2018].

[9] And is not for the faint of heart.

The essence of the methodology lies in the assumption that a data science project is cyclic, and it does not end once a solution is deployed. The process itself can yield new insights and business questions, which extend the problem at hand [Wirth, 2000].

According to CRISP-DM, a project is comprised of six distinct phases. The sequence of these phases is not strict, and cycles between the dependant ones typically occur given a phase's outcome. Due to its flexibility, the methodology is listed as one of the most popular approaches across industries [Kelleher and Tierney, 2018]. As illustrated in Figure 2.4, the six phases are as follows [Wirth, 2000]:

- **Business Understanding:** understanding the project's goals and requirements from a business perspective, and then formulating the problem and a preliminary plan.
- **Data Understanding:** initial data collection and exploration in order to get familiar with the data, identify data quality problems, and take initial insights. This phase is closely related to *Business Understanding*, as defining the problem requires some understanding of the data.
- **Data Preparation:** construction of the final dataset used for modeling purposes. This is likely to be an iterative task throughout a project, and typically involves data cleaning, and attribute selection and creation.
- **Modeling:** selecting and applying multiple modeling techniques, according to the problem type. May require additional data preparation to construct new attributes and further transform data.
- **Evaluation:** once one or more models have been built, more thorough evaluations are needed. This includes reviewing the steps executed to reach this stage to ensure it correctly follows the business objectives.
- **Deployment:** the results are usually organized and presented in a way that is valuable to the customer. Depending on the project, it can be a simple report or the implementation of a complex automated data mining process that results in an application used by the client.

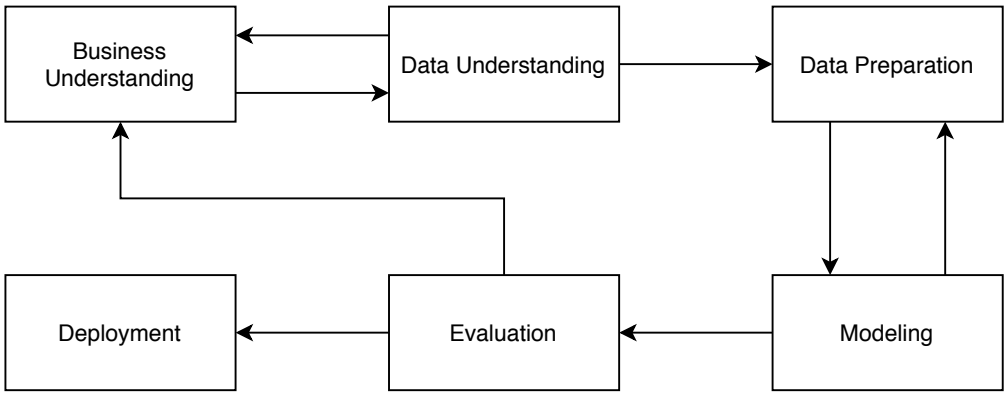


Figure 2.4: Illustration of the CRISP-DM stages.

2.4.2 Machine Learning

Kelleher and Tierney define Machine Learning (ML) as the field of study that develops algorithms that computers follow to identify and extract patterns from data [Kelleher and Tierney, 2018], which are typically employed during the *Modeling* phase of the CRISP-DM process.

Considering this, the majority of ML algorithms can be classified as either *supervised learning* or *unsupervised learning*. The relationship between the two and their respective applications are summarized in Figure 2.5.

- **Supervised Learning**: the goal of supervised learning is to train models to learn the mapping function between an input variable and an output variable, known as the "target" or "response" so that predictions can be made for new input data [Agarwal, 2014]. These algorithms can be applied to either discrete or numerical target variables, defining the problem as either *classification* or *regression*, respectively. For instance, predicting if an email is spam (*yes* or *no*) is a classification problem, while predicting the amount of sales of a future date is a regression problem.

- **Unsupervised Learning:** the main difference of unsupervised learning is that there is no target attribute. The primary task of these algorithms is then to generally discover patterns in the data [Kelleher and Tierney, 2018]. In practice, they are mostly used to look for similar groups of instances, i.e. *clustering*, and to detect dependency rules between instances to identify an item's occurrence based on the presence of other items, i.e., *association*.

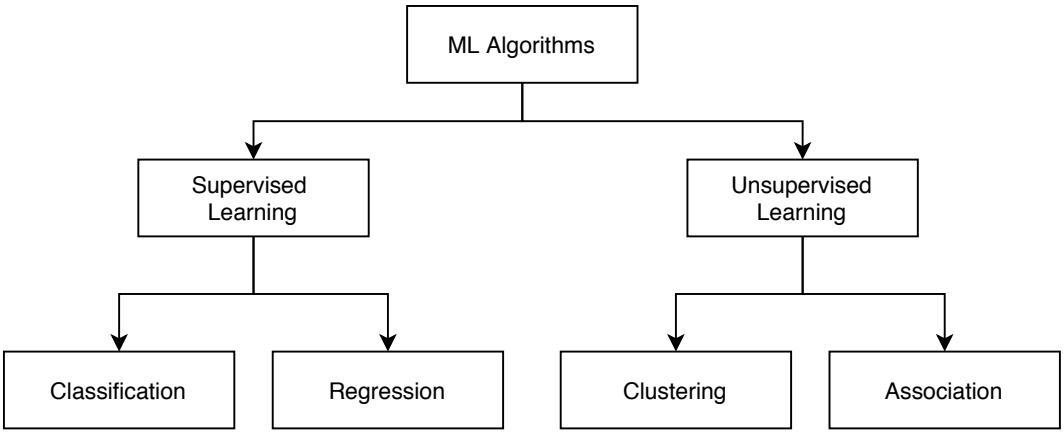


Figure 2.5: Illustration of the grouping of ML algorithms. Adapted from [Garbade, 2018].

2.4.3 Supervised Learning Algorithms

The problem at hand of this dissertation aligns with the goals of supervised learning algorithms, namely, techniques used for regression problems. Considering this, we now review algorithms potentially suitable for our solution.

2.4.3.1 Linear Regression

In its simplest form, Linear Regression models the relationship between a single attribute (X) and the response variable (Y), according to equation 2.7.

$$Y = \beta_0 + \beta_1 X \tag{2.7}$$

In other words, the expected value Y is a straight-line function of X. The parameters (β_i) are selected by the *method of least squares*, i.e., minimizing the squared distance between each Y value and the line of best fit by minimizing the expression 2.8 [Liu et al., 2016].

$$\sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_i))^2 \tag{2.8}$$

Generalizing the approach to model the relationship between more than one predictor (X_i) and the response variable (Y), we get multiple linear regression (equation 2.9).

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + ... + \beta_k X_k \tag{2.9}$$

Modeling phenomenons through linear regression assumes the following principles [Liu et al., 2016]:

- **Linearity:** the relationship betwen X and Y is linear;
- **Homoscedascity:** the variance of the residuals, i.e., the difference between the predicted values and the real values of Y, is the same for any value of X;
- **Independence:** observations are independent of each other;
- **Normality:** for any value of X, Y follows a normal distribution;

Despite its simplicity and assumptions, it is used in multiple applications because of its interpretability and the fact that it can be used to model non-linear relationships, namely by performing *non-linear transformations*, e.g., square root, to either the predictors or response variable.

In supervised learning tasks, overfitting the model to the training data is a potential problem. This is further critical when input data has possibly correlated attributes, hence, not applicable to the method of least squares. Considering this, regularization methods, i.e., extensions to the model which involve adding a penalty term to the error function, can be applied. They have proved effective in reducing overfitting by either dropping or reducing the coefficients of correlated regressors [Brownlee, 2016]. The most common regularization methods include:

- **Lasso Regression (L1):** minimize the absolute sum of the coefficients;
- **Ridge Regression (L2):** minimize the squared absolute sum of the coefficients.

2.4.3.2 Decision Trees

The core idea of the Decision Trees algorithm is a set of *if then, else* rules represented in a binary tree structure [Kelleher and Tierney, 2018]. Even though the original definition was for meant classification, it is proven applicable in the context of regression problems [Brownlee, 2016].

Decision trees are built in a top-down fashion from the root node by partitioning the data that contains entries with similar values, usually measured by entropy or information gain. The result is a binary tree with decision and leaf nodes, where each path from root to leaf defines a classification or regression rule [Kelleher and Tierney, 2018]. Figure 2.6 illustrates a sample tree for predicting a person's gender through their height and width.

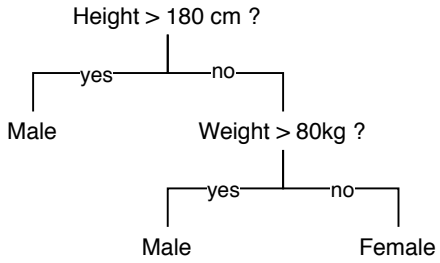


Figure 2.6: Illustration of a decision tree. Adapted from [Brownlee, 2016].

The intuitive formulation of decision trees comes with notable upsides, including [Friedman et al., 2008]:

- **Interpretability:** trees can be visualized, are simple to understand and to interpret;
- **"Off-the-Shelf":** decision trees require little data preparation, unlike other models who need efforts such as data normalization or dummy variables;
- **Whitebox:** model observations can be easily explained by transversing the tree, following boolean logic.

However, decision trees tend to capture noise in the data and overfit, and are highly susceptible to small variations in data. Considering this, they are usually replaced with ensemble methods, e.g., Random Forests (RFs) and Gradient Boosting Machines (GBM), which sacrifice a degree of interpretability in return for higher predictive accuracy.

2.4.3.3 Random Forests

The core idea of bagging algorithms is to average multiple weak but approximately unbiased models, to reduce the variance. Introduced by Breiman, Random Forests (RF) build on the idea of bagging by constructing a large number of de-correlated trees, and then averaging them [Breiman, 2001].

As illustrated in Figure 2.4.3.3, the Random Forests algorithm consists of training multiple decision trees, where each tree considers a random subset of features and training data. In the

context of classification problems, the final output is a majority vote of each tree's output, while in regression problems the results are averaged. The approach is particularly well suited for tree-like regressors, getting accurate results "off-the-shelf", which have turned RFs into one of the most popular supervised learning algorithms [Friedman et al., 2008].

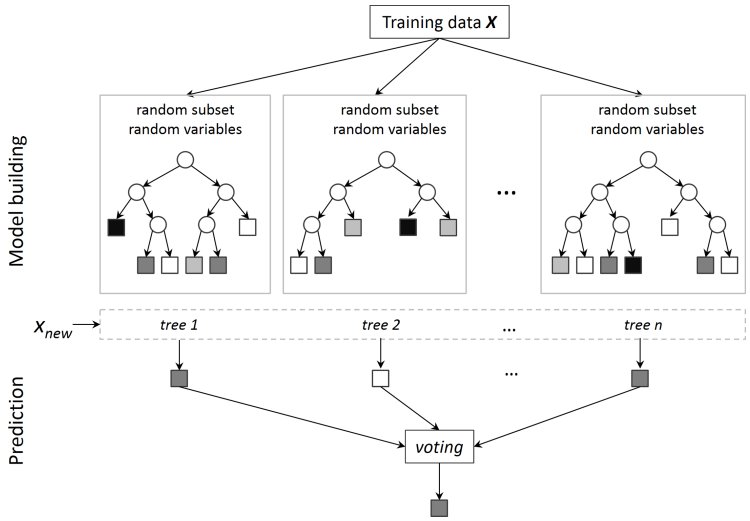


Figure 2.7: Illustration of the Random Forests algorithm for a classification problem [Hu et al., 2016].

Moreover, the algorithm keeps most of the advantages of decision trees algorithms (e.g., interpretability, via *feature importance* analysis), while improving on its flaws: the generalization error for RFs converges as long as the number of trees is large, thus reducing overfitting [Agarwal, 2014].

2.4.3.4 Gradient Boosting Machines

Boosting is similar to bagging in the sense that it thrives in combining outputs from weak learners into a powerful *"committee"* [Friedman et al., 2008]. The difference lies in *how* the learners are combined. Boosting works by training multiple weak learners, where each model in the sequence is fitted giving more importance to observations in the dataset that were poorly handled by the previous learner in the sequence. In other words, latter learners refine the errors of the previous ones.

Boosting is mainly focused on reducing bias. As such, the base learners that are often considered for boosting are models with low variance but high bias, e.g., decision trees. When using trees as the base learners, they are typically shallow, sometimes with a single split.

Introduced by Friedman, Gradient Boosting Machines (GBMs) is a genre of boosting based on decision trees that has proved to be robust in both regression and classification algorithms, especially with noisy data [Friedman, 2001].

2.4.4 AutoML

The modeling phase of the CRISP-DM process involves experimenting with different ML algorithms and their hyperparameters configurations. A model's performance can be further enhanced by combining the outputs of multiple models, to create a "meta-learner" that finds the optimal combinations of the base learners, commonly referred to as Stacking [Breiman, 1996]. Unlike bagging and boosting, the goal of stacking is to ensemble strong, diverse sets of learners, instead of weak, shallow ones. Despite the improvement in predictive capability, this opens a path for further experimentative tasks with potentially little results, time that could be spent in other phases of the CRISP-DM lifecycle.

Considering this, AutoML is an approach at automating all aspects of the machine learning and data analysis pipeline, from data preparation to model deployment. Despite the AutoML effort being in an early stage of this ultimate goal, as of today, numerous AutoML systems that tackle the modeling phase are available, namely by automating model selection, training, tuning, and stacking [Hutter et al., 2019].

These systems have successfully been able to reduce the effort of the modeling phase, while yielding state-of-the-art results in multiple scenarios [Gijsbers et al., 2019, Hutter et al., 2019]. Adhering to the principle of democratizing ML, most of these systems are open-source and based on standard industry tools, e.g., auto-sklearn [Feurer et al., 2015], H2O AutoML [H2O.ai, 2017], and Auto-WEKA [Kotthoff et al., 2017].

2.5 Summary

Retail is a sophisticated environment. The emergence of transactional data and computing power of the last decades motivated computerized approaches to its numerous challenges, including sales forecasting, PED estimation, and MDO.

Some form of sales forecasting is crucial in guaranteeing a retailer's performance, given most decisions, both operational and administrative, depend on an estimate of future events. This has intensified the study on the factors that influence demand, namely the ones that can be modeled through time series. However, the recent trends in retail sales, characterized by an increased number of new products of short life cycles and erratic demand, limit the traditional approaches' feasibility when forecasting for operational decisions such as pricing. We expect that using more advanced non-parametric techniques will be suitable for the predictive tasks at hand.

Moreover, retailers have long identified price as a significant factor in driving demand, which has motivated the study of the relationship between the two. This measure, PED, has been used in multiple industries to support pricing decisions. In the context of retail, it is primarily employed in MDO scenarios, most of which formulate demand via parametric regressions and assume power or exponential relationship between price and demand. When more complex formulations are needed, e.g., non-parametric regressors, PED is indirectly estimated via other price related features.

Traditionally, price markdown decisions under clearance sales are made using "rule-of-thumb" techniques according to the managers' experience. However, in recent years, computerized approaches have emerged, namely through the field of MDO. The approach reached multiple successful studies, with some of which turning into commercial tools. However, most retailers still operate under the traditional setting, despite the manual effort involved and the questionable results. Such is the case of the retailer under study in this work, which is nowhere near an automated solution for setting price markdowns. Nevertheless, the available commercial tools do not have the flexibility required for seamless integration within not just the managers' workflow, but also the retailers' information system. Considering this, a bespoke approach is needed, hence the motivation for the project under which this dissertation arises, a decision further supported by our review of the literature.

To successfully achieve our goals, we deal with the multiple challenges associated with working on data from a retailer of this magnitude and the infamous inaccuracy of inventory information systems. Following the CRISP-DM process, we expect to increase the chances of this work's success.

Chapter 3

Problem Statement

This chapter describes in detail the problem at hand and the proposed solution. Section 3.1 introduces the retailer target of this work. Section 3.2 lists the main issues faced by the retailer in the context of price markdown decisions. Section 3.3 lists the primary goals of the project this dissertation arises in, highlighting the ones this work focuses on. Section 3.4 extends the previous one, listing both functional and non-functional requirements that condition this work. Section 3.5 provides an overview of the various components of the solution, and lists the tools used to achieve them. Section 3.6 summarizes the points mentioned above.

3.1 Retailer Overview

This work's target company is a national leader in retail. As of 2018, it had over 300 stores throughout mainland Portugal, and 30% of its sales volume is attributed to own-brand items [Barbosa, 2018].

Product-wise, the retailer is split into three main *Divisions*, namely *Food*, *Fresh*, and *Non-Food*. These comprise several *Business Units*, which branch further (e.g., into *Categories, Subcategories*) until reaching the individual Stock Keeping Unit (SKU), i.e., a product. Figure 3.1 illustrates an overview of this arrangement.

In this light, our focus is on the company's *Non-Food* division, which holds the retailers' non-edible items. More specifically, three categories were selected to be part of this project, namely[1]:

[1]The category names are illustrative, for confidentiality reasons.

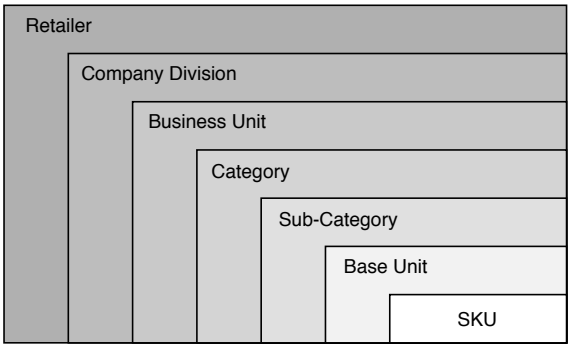


Figure 3.1: The retailer's product hierarchy.

- **Home:** multiple types of home appliance and decoration items (e.g., candles, cooking supplies, lamps);
- **Office:** mainly comprised of stationery items (e.g., notebooks, office supplies);
- **Textile:** the retailer's own-brand apparel items (e.g., t-shirts, shoes).

Figure 3.2 illustrates the positioning of the Non-Food division and the target categories within the retailer's structure. For simplicity, we skip the *Business Unit level*.

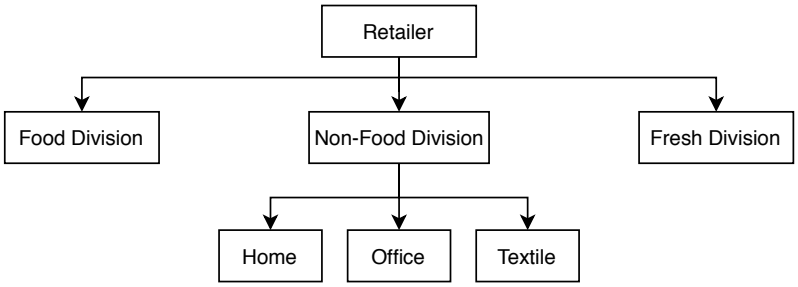


Figure 3.2: The target categories within the retailer's structure.

Additionally, the retailer has different store formats. These formats differ in their size, and consequently, the type of items sold. We consider the three main store formats [Barbosa, 2018]:

- **Format A:** Large stores with a high range of products and sales volume. Usually located in densely populated areas;
- **Format B:** Medium-sized stores. Usually located outside cities;
- **Format C:** Smaller stores with a short product range. Located to serve specific populations.

Within the last year, the target categories had over thirty thousand different active SKUs. This leads to numerous sales pattern types: some products are permanently in the stores and have recurrent stock replenishments, while others are purchased for a single season and are not expected to make a return. In an attempt to standardize the variety of business goals, the retailer categorized the products by their *nature*:

- **Permanent**: items permanently sold all year round, with no foreseen end of life. Usually the "basic" versions of items which have demand all year round, e.g., simple notebooks, plain t-shirts.
- **In and Out**: items sold at specific seasons according to known seasonal trends. These items are only sold in a predefined time-window, but the products are expected to make a return to the stores in the future (e.g., Christmas trees, which stop selling after Christmas but return the following year).
- **One and Done**: items purchased for a specific season given current trends, or as an experiment by managers. These have a predefined end of life, and no additional stock is purchased.

3.2 Current Issues

When it comes to monitoring sales performance and making replenishment and pricing decisions, the retailer follows a traditional approach, with "rule-of-thumb" strategies according to the managers' experience and known seasonalities. There is no standardized and analytically sound reasoning behind price markdowns and related decisions. Besides the questionable quality of this approach, the process itself is rather inefficient: each week, managers have to single out which products will be the target of price changes, usually from a pool of hundreds of active items across stores. During this process, they have access to raw metrics, such as daily sales and stock values. From this shared standpoint, each manager and respective team employ their own analysis with their preferred tools and bespoke metrics, which turn into operational decisions, e.g., pricing, replenishment stock purchasing.

This method, profoundly dependant on human resources, does come with an upside: specialized managers who have deep experience in their products' characteristics and nuances. However, given the scale of the company, the ever-changing environment of retail, the heterogeneity of store formats, product types, and sales patterns make this approach unsustainable, both from an operational and quality standpoint.

3.3 Goals and Scope

As introduced in Chapter 1, the retailer proposed an internal project carried out by LTPlabs, under which this dissertation arises. The project has the primary goal of improving product life cycle management and price markdown strategies through computerized assistance. In particular, the major goals of the project include:

- **Improving Product Life Cycle Management** by (i) developing a methodology to forecast product sales, with an emphasis on predicting sales performance in the later stages of the products' life cycle; (ii) identifying underperforming products so that corrective measures can be applied, e.g., a price markdown.

- **Supporting Price Markdown Strategy** by developing a methodology that predicts the best price markdown for underperforming products.
- **Automate Outlier Detection*** by calculating a set of relevant business metrics for each product, e.g., "average stock per store", and highlight outliers, i.e., values too high or too low.
- **Improve Decision Efficiency*** by developing a DSS that incorporates the above items, allowing retail managers to interact, analyze, and execute operational decisions, e.g. submitting a price change.

This dissertation focuses solely on the first two goals, i.e., items marked with * are out of scope of this work, but are relevant to mention because of their inherent influence on the presented approach.

3.4 Requirements

Given the goals listed above, here we present the main functional requirements of this work, which should be integrated within the DSS:

- **Sales Forecasting:** provide a standardized sales forecasting system for medium-term horizons, at a product level. It should also work for new items, i.e., with no historical data;
- **Underperforming Item Detection:** detect which products are underperforming, i.e., will not meet their stockout targets by end of life;
- **"What-If" Analysis for Price Markdowns:** simulate the impact of discrete price changes (e.g., 10%, 20% off) on product demand and related metrics;
- **Price Markdown Optimization:** for underperforming products, suggest the optimal price markdown that ensures stockout targets are met;

Additionally, we deal with a number of non-functional requirements, which constrain how the functional requirements are accomplished:

- **Scalability:** the solution will be extended beyond the scope of the project and respective target categories. In particular, it should be prepared to accommodate up to 50 categories. The size of the remaining categories is comparable to the current ones, which would mean a 17 fold increase in terms of data footprint, if all categories are integrated;
- **Explainability:** the end-users of the DSS are retail managers. White-box machine learning models (e.g., decision trees, linear regressions) should be favored, so the results are easily interpreted even by people with no prior knowledge on regression techniques.

3.5 Proposed Solution

To tackle the issues described in Section 3.2, and given the requirements listed in Section 3.4, our solution is three-fold, followed by an integration phase.

Sales Forecasting Model The first component is a forecasting model that predicts weekly sales for each product across all stores, until the end of each product's life cycle[2]. We use supervised learning methods, namely tree-based regressors, to model with additional predictors based on the factors that influence demand (see Section 2.1.3), e.g., seasonality and the inventory effect. Additionally, we use k-means clustering to group products with similar life cycle profiles, tackling sets of related seasons individually.

The models are trained and predict aggregated weekly values, but consider inventory constrains according to each store's weight on total sales values. Furthermore, we train a classification model to assign new products (no historical data) to an existing cluster, according to their attributes to ensure relevant predictions. The forecasts are used to compute estimates of cumulative sold stock by the end of season, based on which we identify underperforming products. Given the difference in sales behavior and assortment strategies, the approach is followed independently per category. Figure 3.3 illustrates the high-level solution.

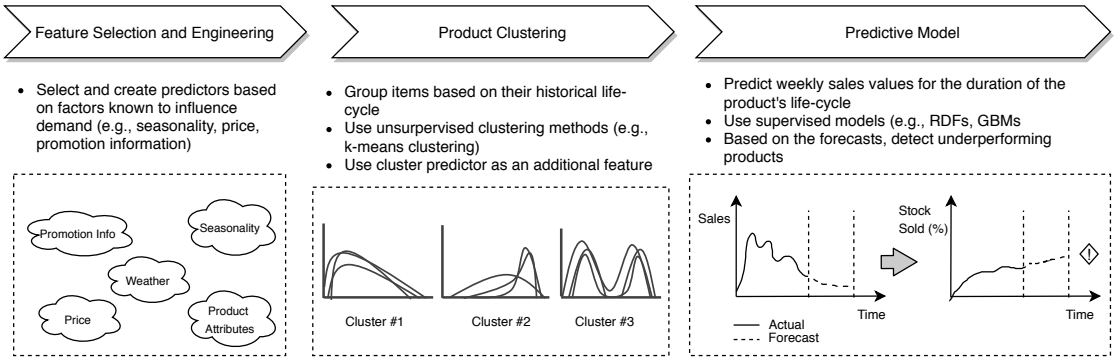


Figure 3.3: Overview of the Sales Forecasting solution.

PED Estimation Model The second component is a price elasticity of demand estimation model. Its main goal is to predict the demand increase of a product in response to a price reduction, e.g., predict the impact of a 20% discount on a product's sales volume. To better retain the price factor, we filter data to only include weeks where price changes occurred, labeled with the corresponding sales volume variation of the following week. We then follow standard feature engineering practices to train a supervised model. We privilege modeling information related to price (e.g., the applied discount, price of similar products), the effect of time, (e.g., product age, seasonality), and product characteristics. Similarly to the sales

[2]Or up to a year if the product is permanently in-store

forecasting procedure, we approach each category independently. Figure 3.4 illustrates the high-level solution, of which we take an in-depth look in Chapter 4.

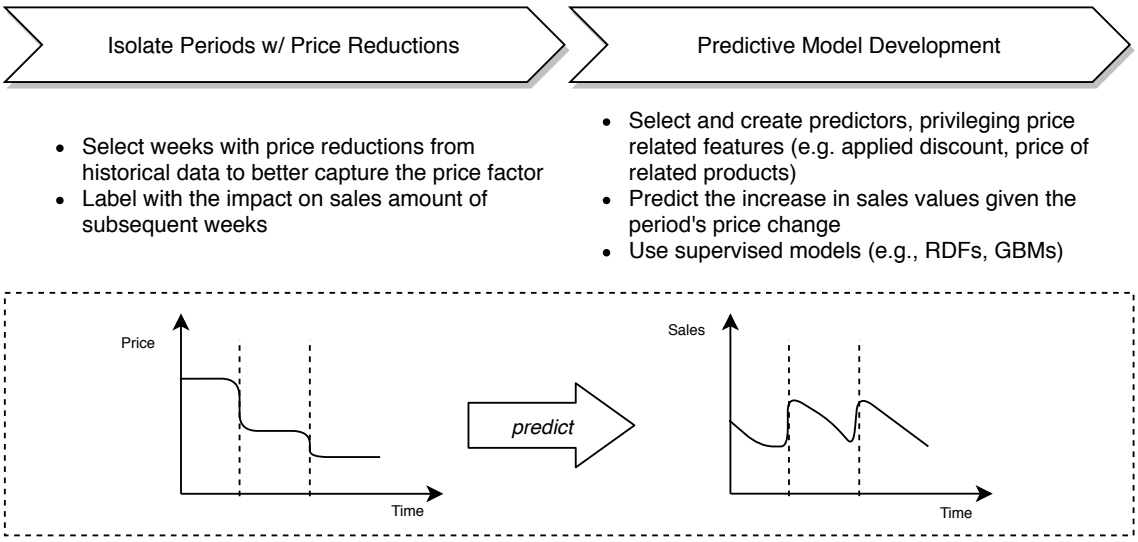


Figure 3.4: Overview of the PED estimation solution.

Markdown Optimization Finally, we formulate a price markdown optimization problem. Based on the sales forecasts, we identify products that will not meet their stockout targets by the end of life. For these items, we combine the original forecasts with the PED estimations to predict new weekly forecasts given the expected impact of applying discrete price changes (e.g., 20%, 50% off). The goal is to ensure minimum stockout targets are met while maximizing future cumulative margin. Considering this, we compute the optimal price markdown for each underperforming product. We make use of the remaining markdown computations to display impact simulations of applying each markdown. Once again, figure 3.5 illustrates a general approach - we will delve further into this topic on Chapter 4.

Lastly, we integrate the components within a DSS developed in parallel to this work, which, namely the sales forecasts, price markdown simulations, and optimal price markdown. The system has weekly updates with new data, sales forecasts, and price simulations. To achieve this, we build an automated pipeline that ensures all stages of the deployment process, i.e., data extraction, data processing, model training and application, and price markdown optimization. Figure 3.6 illustrates an example data workflow of the solution.

To further validate our solution, we run a pilot test lasting four weeks where the DSS's performance is evaluated alongside its users. It is expected that the feedback from the managers will allow us to iteratively improve the solution, namely the performance of the use cases developed by this work.

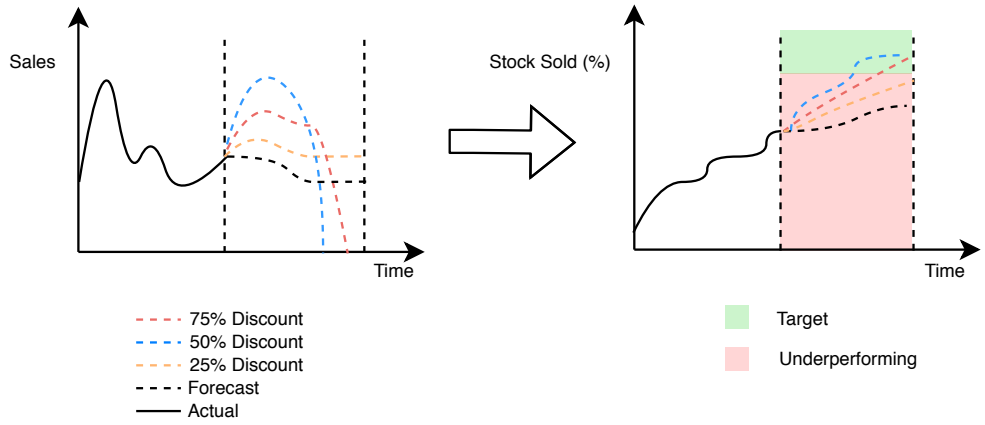


Figure 3.5: Illustration of the price markdown optimization approach. By simulating the impact of different price changes (e.g., 25%, 50%, 75% discount), we look to compute the price strategy that ensures stockout targets are met (in this example, 50% discount).

3.5.1 Tools used

The infrastructure in place in the retailer constrained the tools used throughout the project, and the final architecture of the end solution. In particular, the major components in use were:

- **Apache Hadoop:** the storage solution for daily operational data and main data source for our work. Apache Hadoop[3] is a framework for distributed processing of large datasets across clusters of computers. It relies on the Hadoop Distributed File System (HDFS) for high-throughput data access.

- **Apache Impala:** the main access point to data. Apache Impala is a SQL Query engine integrated within the Hadoop environment. It is the highest performing SQL-on-Hadoop system, being on average 18 times faster than alternatives, e.g., Apache Hive[4] and Presto[5] [Kornacker et al., 2015].

[3] https://hadoop.apache.org/
[4] https://hive.apache.org/
[5] https://prestodb.io/

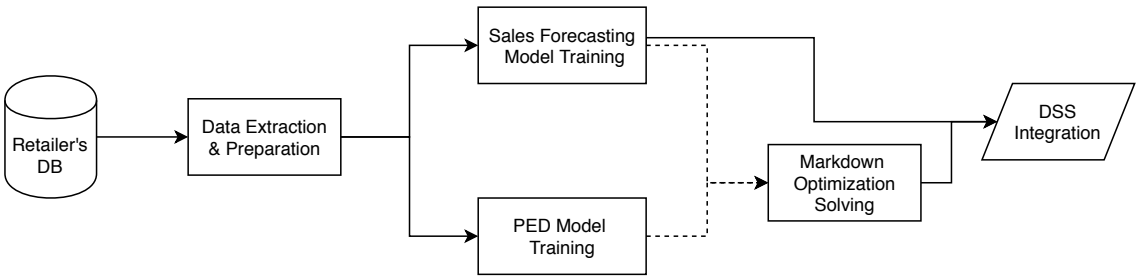


Figure 3.6: Expected data flow of the final solution.

- **Cloudera Data Science Workbench (CDSW):** the production environment for data science related tasks. CDSW[6] is a platform for collaborative data science projects for enterprise systems, which can run arbitrary Python, R, and Scala projects. We use it as our production environment and use its orchestration features to build our recurrent pipeline.

- **H2O:** framework used for the development of ML models. H2O[7] is an open-source machine learning platform featuring the most widely used statistical and machine learning algorithms (e.g., gradient boosting machines, generalized linear models). We chose it over common alternatives (e.g. scikit-learn[8]) because of its better scalability, and mature model deployment capabilities, which are crucial for the work's success [Pafka, 2019, Saberidokht, 2019].

3.6 Summary

This work aims to improve a large Portuguese retailer's price markdown strategy from a quality and efficiency standpoint, with a focus on the late stages of the product life cycle. In particular, we focus on three target categories from the *Non-Food* division, which feature substantially different types of products. However, the solution should be as category-agnostic as possible, so it can be extended to the remainder of the retailer's division.

With the project, we expect to achieve our goals by delivering a DSS that empowers the categories' managers with a number of use cases that support their operational decisions. Considering this, our work aims to design and develop a methodology for the use cases related to sales forecasting and price markdown decisions.

To do so, we propose a three-fold solution: a sales forecasting model with the primary goal of identifying underperforming products; a PED estimation model that predicts the amount of sales increase in response to a given price markdown; finally, using both models, we compute price markdown scenarios from a set of discrete options (e.g., 10%, 25%) from which we yield the optimal price markdown.

With our solution, we expect retail managers to have direct and automated access to the products in need of markdowns at a given time, as well as the optimal price to apply to each. If additional control is needed, simulations on the impact of different prices can be explored. As mentioned in Section 1.2.2, we evaluate this work's success by improving the current approach at sales forecasting and price markdown quality, evaluated by comparing historical data. Furthermore, we evaluate the acceptance level of the solution by running a pilot test alongside retail managers.

[6] https://www.cloudera.com/products/data-science-and-engineering/data-science-workbench.html
[7] https://www.h2o.ai/
[8] https://scikit-learn.org/

Chapter 4

Implementation

This chapter takes an in-depth look at the solution presented in the previous chapter, details the decisions made to reach it, and describes essential implementation steps. In particular, Section 4.1 describes the data available for our work, its issues and how we solved them, followed by a set of explorations. Section 4.2 describes in detail the implementation of the sales forecasting model, including the used predictors, the product clustering solution and the approach for predicting new items. Similarly, Section 4.3 thoroughly explains the reasoning behind the PED estimation model formulation, followed by its implementation steps. Section 4.5 presents the architecture and flow of the deployment solution, and overviews the pilot test. Finally, Section 4.6 summarizes the whole process.

4.1 Data Analysis

The first challenge during this dissertation's work was to explore, clean, and process the available data to fit our needs, both for model development and integration within the DSS. It was an ongoing process throughout the entire work and significantly contributed to the quality of the final solution.

4.1.1 Data Access

Throughout our work, we had access to the retailer's data through a Hadoop cluster with data from as early as 2016. The data was spread across multiple tables with different information,

e.g., product characteristics and stock amounts. It is an operational data source, meaning most time-series data was recorded daily, and for each SKU x Location[1] combination.

We accessed the data via SQL queries to the cluster, whose results could be downloaded and further explored offline if needed. Early explorations revealed the multiple challenges associated with working with this data source:

- **Scale:** being a large retailer with hundreds of stores and thousands of products, some tables have dozens of millions of rows.
- **Complexity:** the data source's primary goal is to ensure daily operations across the supply chain. Over the years, it has become a complex system, featuring numerous tables with undocumented relationships.
- **Data Consistency:** missing data, wrong values, and inconsistent connections between tables are frequent. Inaccuracy is typical of retail information systems [Nachtmann et al., 2010, Kang and Gershwin, 2005], and this one was no exception. Additionally, crucial knowledge was not inferable, e.g., if a given product had stock in a store, we could not determine if it was being sold (i.e., on store shelves), or sitting in the store warehouse.

Before any serious usage, a thorough understanding of the intricacies of the data source was required. Therefore, data improvement was an ongoing process throughout our work. It took multiple interactions with retail managers and IT professionals to ensure that the data we were accessing or transforming was correct.

4.1.2 Data Preparation

The available data was not applicable for our use cases. For instance, and as previously mentioned, all original time-series data was recorded at a daily level, and for each individual location. However, our solution envisioned weekly forecasts across all stores, which requires data to be aggregated by time and location. Additionally, the data was spread across multiple tables, e.g., a table for sales data, one for product information, another for stock amounts. This is a typical architecture of Online Transaction Processes (OLTP), but it is not suitable for data science related tasks. This requires a data structure that facilitates analysis, e.g., with a more straightforward data schema, no normalization, and aggregated data.

To overcome this, we designed a new set of table schemas and a procedure to transform the data into its final form, akin to an Extract, Transform, Load (ETL) procedure. Initial attempts were made to implement these transformations via offline Python scripts, to bridge the gap between data preparation and model development. However, we quickly realized the scale of the data required a more robust approach. We then implemented a set of SQL scripts that run directly on the clusters' tables and create a workable dataset, which we iteratively improved for performance optimization.

[1] A Location is a typically a store or a warehouse.

In general, this procedure did the following: (i) filter data - we were only interested in items from the target categories; (ii) aggregate data by time and location - we wanted weekly, not daily information; (iii) join and clean fields spread across multiple tables; (iv) calculate new fields to be used as model features.

Another major issue with the data was noise. For products with in-store stock and zero sales, we could not differentiate between cases of no demand and products that were not on display. This happens because of leftover stock kept in-store or in warehouses for later use, e.g., keeping Christmas trees for the following year, or not used at all. The issue affected a significant amount of data points, and keeping this noise would negatively impact model performance. To fix it, we designed a system where we divide a year into commonly used selling periods - *Seasons*. Each Season is a time-slot ranging from a month, e.g., *Halloween*, to the entire year, i.e. *Permanent*. We then worked alongside managers to match products into each *Season*, for all available historical data. By filtering out data points outside these slots, we substantially reduced data noise. Additionally, this allowed us to know the starting selling date(s) and target outdate(s) for each product. Figure 4.1 illustrates the usage of the *Seasons* system, listing the most relevant ones.

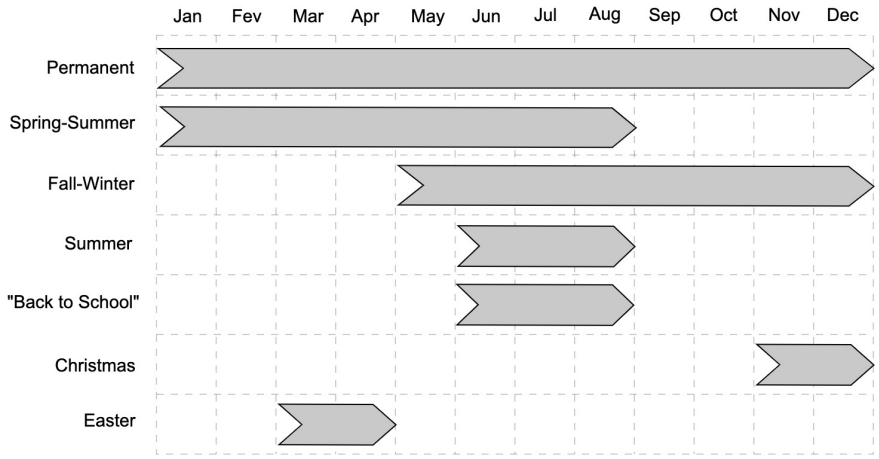


Figure 4.1: Illustration of the *Seasons* allocation system. We assigned each product to their respective selling *Seasons* for all historical data. Some *Seasons* can overlap or even have the same period, e.g., *Summer* and *"Back to School"*, because they hold different types of products. Each year has 16 Seasons that repeat, e.g., *Christmas18* and *Christmas19*.

Furthermore, it was a challenge to obtain consistent price information. The retailer's loyalty program (which may offer personalized discounts), combined with price differences between stores, made relying on individual transaction prices an unfeasible approach. To work around this, we computed the weekly aggregated revenue and the number of sales, and obtained price by dividing them:

$$Price = \frac{Weekly\ Revenue}{\#\,Weekly\ Sales}$$

Table 4.1: Summary of available the data after preparation.

Type	Name	Aggregation	Description
Dynamic	Sales	SKU x Week	Weekly sales volume
	Stock	SKU x Week	Weekly stock amount
	Price	SKU x Week	Weekly selling price
	Active Stores	SKU x Week	Number of stores with available stock
	Selling Window	SKU x Season	Start and end date of the current *Season*
Fixed	Product Hierarchy	SKU	Hierarchy information, e.g., subcategory (see Fig. 3.1)
	Others	SKU	Other SKU information, e.g., description, color, size

We calculated this for each SKU x Week x Store combination, i.e., for each product, the weekly price per store. To obtain a single price per SKU x Week, we chose the most common price across stores, at each week. This holds low risk because it is the retailer's policy to normalize prices across stores as much as possible, meaning only a few stores have different prices each week. In weeks with no sales (*Price* is zero), we employed temporal forward filling, i.e., using the most recent valid entry.

Finally, we retrieved SKU related entries, namely relevant hierarchy information, i.e., category sub-category, and base-unit (see Fig. 3.1), and other characteristics (if available), e.g., color, description, and size. Table 4.1 summarizes the data after transformation, which was the starting dataset for subsequent model development.

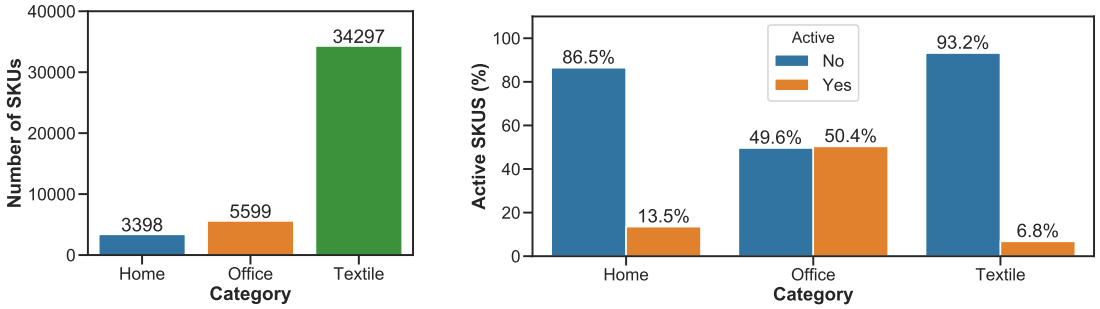
4.1.3 Data Exploration

At this point we were confident the data we were accessing was more reliable, especially for the target categories. Considering this, we proceeded with an exploratory analysis.

In total, between 2016 and June 2020, the three categories comprised over forty thousand SKUs, with *Textile* holding most of the items. This is in part because most apparel items have multiple sizes, each registered as a unique SKU. Figure 4.2a shows the total amount of products of each target category.

Naturally, not all of these items are currently in-store. While some have been deprecated over time, others are on hold until the return of a valid *Season*. The *Office* category holds the largest relative amount of active products, with half of their total SKUs currently in-store. Figure 4.2b shows the relative amount of active SKUs of each target category.

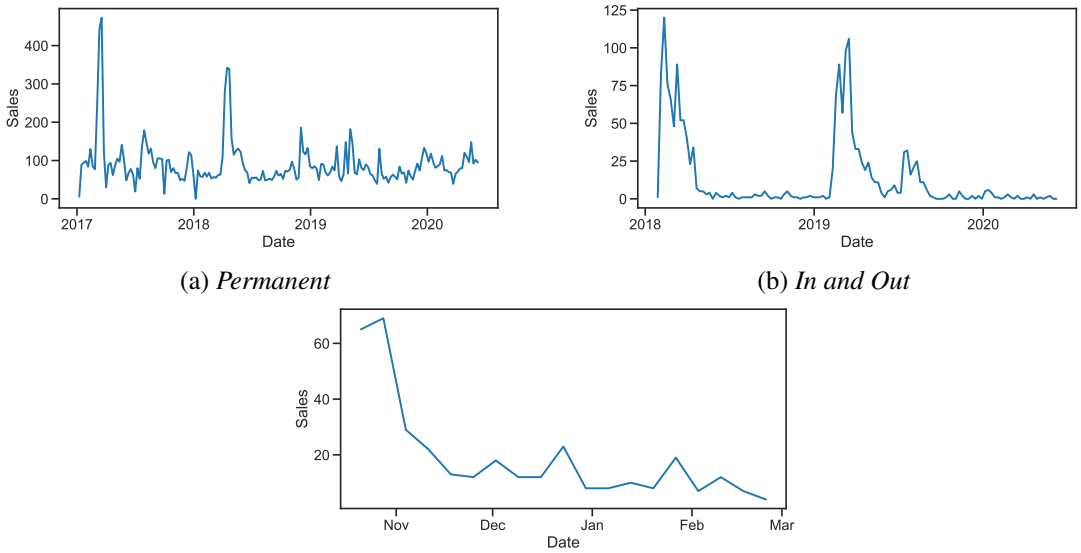
Additionally, the *Seasons*-based data cleanup allowed us to identify the periods where items were in store. This helped us confirm the typical sales pattern of the different product natures. As displayed in Figure 4.3, *Permanent* items hold long historical data with steady sales given the perpetual store exposure (Fig. 4.3a). *In and Out* products have seasonal patterns with sales spikes during the respective in-seasons and no sales during the rest of the period, with a return the following year (Fig. 4.3b). Similarly to the example, some products have residual sales during the off-season due to left-over stock. Finally, *One and Done* products feature a short life cycle (typically 2 to 6 months), with no return to stores, hence, there is little historical data (Fig. 4.3c).



(a) Amount of SKUs per target category.

(b) Relative amount of active SKUs, per category.

Figure 4.2: Illustrations of SKU distribution among the target categories, i.e., *Home*, *Office*, and *Textile*, and active state.



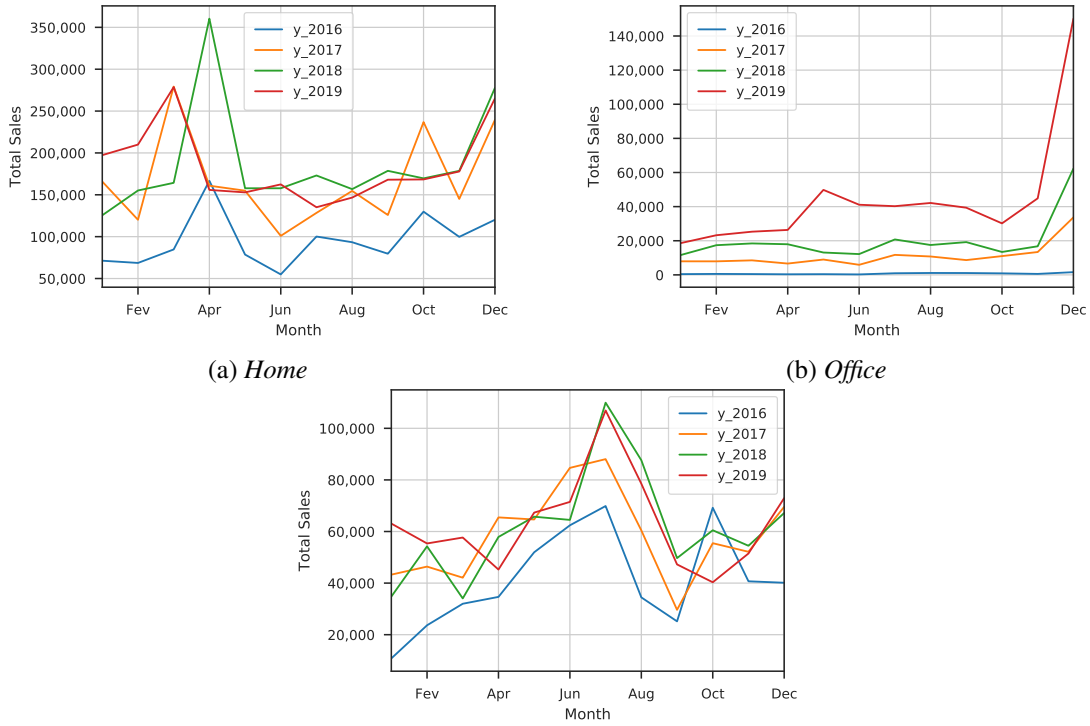
(a) *Permanent*

(b) *In and Out*

(c) *One and Done*

Figure 4.3: Sales history of sample SKUs from different product natures.

Furthermore, we assessed the sales volume of each category to detect potential seasonalities. Figure 4.4 displays the total monthly sales (in units) of each category between 2016 and 2019. All categories show a general increase in sales after 2016, with the remaining years showing comparable performance. *Home* (Fig. 5.1a) is characterized by a sales peaks around April and December, and is the category with the highest sales volume. *Office* (Fig. 5.1b) shows a consistent year-on-year increase in sales with a peak at the end of the year. *Textile* (Fig. 5.1c) shows a sales peak in the Summer, namely in August, and a minor spike by December. However, despite having six times more SKUs (see Fig. 4.2a), its sales volume is comparable to that of *Office* in 2019.



(a) *Home* (b) *Office*

(c) *Textile*

Figure 4.4: Evolution of monthly sales for each target category, between 2016 and 2019.

4.2 Forecast Model Development

After constructing the dataset with the relevant data, we focused on the development of the key module of the solution, a sales forecasting model. As mentioned in Section 3.5, the goal is to forecast weekly sales for each SKU, including new items. The primary functional usage of the forecasts is to identify underperforming products, so pricing decisions can be taken.

4.2.1 Defining Predictors

Determining potential predictors that could be derived from the data was an iterative process, driven mostly by: (i) new consolidated data points (e.g., promotion information was only available later during development) (ii) retailer manager's feedback on predictions for specific products (iii) validation and testing sets results.

The characteristics of the data made forecasting weekly demand for each SKU a challenging task, namely given the range of historical data available per SKU. For instance, *Permanent* products have data spanning multiple years, while some *One and Done* items stay on sale for just two months. This made it unfeasible to rely on forecasting methods that are good at modeling seasonality but require long historical data (e.g., ARIMA). Nevertheless, because our goal is to create weekly forecasts for each SKU's entire life cycle (even for new products), we needed to capture the long term sales behavior across all product types.

As mentioned in Section 3.5, the approach should be category-agnostic so that it can be reproduced for other categories other than the ones targeted by this work. This limited us to using information available across all categories, making it impossible to use some SKU information, e.g., color and size, data that is only available for *Textile* and are popular predictors for fashion items [Thomassey, 2014].

Given these constraints and the literature review on factors typically used to model demand, we created and experimented with multiple predictor combinations. After extensive experimentation, we now list the final predictors included in the model, with the respective context for the less obvious ones:

- **Price:** the SKU's weekly price, computed as mentioned in Section 4.1.2;
- **Stock:** the amount of stock available in-store. It is well known that an article's demand declines when inventory goes below a certain level, and it's a common factor in demand function definitions (e.g. [Smith and Achabal, 1998, Vakhutinsky et al., 2012]) . We chose not to use the broken assortment effect formulation (see Section 2.1.3), and use direct stock amounts instead, because it is commonly applied to apparel items only (e.g. [Caro and Gallien, 2012]), and would increase predictor complexity;
- **Current Year Performance:** the rolling mean of weekly sales of the last twelve months, or the entire history if the product has less than one year of life.
- **Recent Performance:** the rolling mean of weekly sales of the previous four weeks. A product's demand showed some degree of autocorrelation, but using just the previous week's demand was too susceptible to outliers. Using the last four weeks presented a good fit while still being an interpretable predictor;
- **Homologous Period Performance:** weekly *sales factor* of the homologous period, i.e., for *Permanent* products, the corresponding week in the previous year, for *In and Out* products, the corresponding week in the previous *Season* it was on sale. A *sales factor* represents an item's performance relative to its historical data, a measure we take an in-depth look in Section 4.2.3;
- **Similar Products Performance:** weekly seasonality index of similar products for the last 12 months, here represented by SKUs within the same *Base Unit*, e.g., "men's summer shorts". It is highly representative of the seasonality of a product group across the year, and helps predict demand spikes. Computed via additive seasonality decomposition using moving averages [Hyndman and Athanasopoulos, 2018];
- **Product Age:** number of weeks since the SKU has been introduced to the stores, i.e., the beginning of the corresponding *Season*. For *Permanent* products, the number of weeks since the last major stock replenishment, i.e., an increase of at least 50% in stock;

- **Cluster Prediction:** the weekly sales prediction of the cluster the SKU belongs to. It is the most important predictor and the main novelty of the approach. We take an in-depth look at the clustering approach in Section 4.2.3;

- **Number of Active Stores:** because we are predicting sales across all stores, we risk losing effects only captured with less aggregated data. Therefore, the goal of this predictor is to capture two effects: the impact of sales as stores begin to stockout, and widespread product availability - some products are only available in a given store format (see Section 3.1);

- **Seasonality Dummies:** dummy variables to capture additional quarterly and yearly seasonality at the SKU level, a standard approach at modeling seasonality [Hyndman and Athanasopoulos, 2018];

- **Promotion Information:** boolean indicating if the SKU is part of the promotion roster that week. We intend to capture the *display effect*: when products are under promotion, they feature additional in-store advertisement, which is known to increase demand [Ailawadi et al., 2006].

Before establishing the listed predictors, we experimented with multiple attributes that did not make it to the final modulation. Honorable mentions include: ABC classification, i.e., a system based on the Pareto principle used to distinguish the most sold items (A), from the remaining ones (B and C), and that have been used for sales forecasting [Stellwagen, 2013]; hierarchy information, i.e., adding the sub-category or base unit the SKU belongs to. Being static information, they did not contribute to the model's predictive power, yielding residual feature importance. Also, both would be categorical predictors, which contribute further to feature explosion when encoded via dummy variables.

4.2.2 Modeling Specifications

There is no best Machine Learning (ML) algorithm that, on average, outperforms all others across all situations, i.e., "*no free lunch*" [Wolpert and Macready, 1997]. In this sense, it is essential to test with multiple modeling algorithms and compare results before choosing a final model.

Considering this, we ran multiple experiments with different supervised learning algorithms. As mentioned in Section 1.2, we favored whitebox ML models so results can be easily interpreted by retail managers, who typically have no prior knowledge on regression techniques. Moreover, multiple works suggest that explainability increases analytical systems' adoptance [Ferreira et al., 2016, Kelleher and Tierney, 2018]. We then focused on tree-based algorithms because of their intuitive structure and visual representation of feature importance. As mentioned in Section 2.4.3.2, another upside of this type of regressors is that they do not require feature normalization. More specifically, we experimented with the following algorithms:

1. Random Forests (RF);

2. Extremely Randomized Trees (XRT);

3. Gradient Boosting Machines (GBM).

Additionally, we performed hyperparameter tuning within each model's possible parameters. Since we chose tree-based regressors, they have multiple parameters in common, while others are specific to GBMs. We now list the parameters tuned, under H2O's Python API format [H2O.ai, 2020]:

- *n_estimators*: the number of trees in the regressor;
- *max_depth* the maximum tree depth. Higher values make the model more complex and may lead to overfitting.
- *sample_rate*: the row sampling rate. Higher values may improve training accuracy; test accuracy improves when either columns or rows are sampled [Friedman, 2002].
- *learn_rate*: the rate at which a GBM learns when building a model. Lower learning rates are generally better, but they require more trees.

A standard procedure at hyperparameter optimization is to perform Grid Search, where a range of parameters are specified and a model is trained for each possible combination. Random Grid Search is randomly selecting a subset of the options to avoid searching over the entire space. Following this procedure, we limited the search to a total of 250 models. We performed the described search individually for each algorithm. Listing 4.1 shows the specified hyperparameters of each search, under the format of H20's Python API [H2O.ai, 2020].

```
hyper_params = {
        'learn_rate' : [i for i in range(0.05, 0.3, 0.05)],
        'max_depth' : [i for i in range(2, 8, 1)],
        'ntrees' : [i for i in range(20, 200, 10)],
        'sample_rate' : [i for i in range(0.5, 1.0, 0.1)],
        'col_sample_rate' : [i for i in range(0.3, 1.1, 0.1)],
}
```

Listing 4.1: Hyperparameters specified upon grid search.

Given that the problem formulation is time-dependant, we must be cautious with the use of cross validation (CV) because of the inherent serial correlation of time series data. Under this condition, the typical approach is to perform out-of-sample evaluation, i.e., with a train-test split [Bergmeir et al., 2018][2]. Considering this, we evaluate the results through the use of a train and validation set (last 20% of the training data). We then ranked the models by the resulting Mean Squared Error (MSE), and compared the results of each algorithm's best model.

[2]One exception is in the case of purely autoregressive models, which is not the case. [Hyndman and Athanasopoulos, 2018]

The COVID-19 pandemic outbreak had its early stages during the modeling and evaluation phase of this dissertation, which has dramatically affected all economic sectors, including retail [Fernandes, 2020]. Taking this into account, we did not consider 2020 data while evaluating model performance[3]. Furthermore, the functional goal of the forecasting model was to assist in setting price markdowns, which are typically applied late in the *Season*. Hence, our training and validation set ranged from 2016 to April 2019, with the remaining data being held out for further analysis, namely the last 16 weeks of the longest running *Season* of 2019, i.e., *Spring-Summer*. We take an in-depth look at the results in Chapter 5.

4.2.3 Clustering Products

One of the main challenges of the data was that most non-*Permanent* products have a short life cycle, sometimes under 2 months. Regardless of this short span, the life cycle curve of products varied highly: while some items had their performance peak in early life, i.e., right after starting to sell (Fig. 4.5a), others performed their best in their final weeks in-store (Fig. 4.5b). Some even showed two separate performance peaks within a single *Season* (Fig. 4.5c).

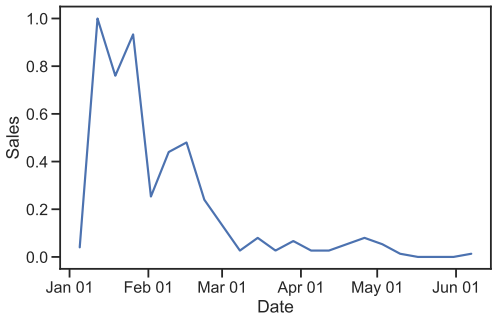
These different life cycle profiles were a significant challenge when forecasting sales over medium to long term horizons, because early sales performance may not be representative of the product's overall demand, e.g., Fig. 4.5b. Moreover, the large number of products of diverse types indicated that other, less intuitive life cycle patterns were present, which were not feasible to detect by manual data exploration.

Considering this, our solution was to identify products with related life cycle profiles via a clustering algorithm. In fact, aggregating sales from items with similar sales patterns have shown to reduce prediction variability [Demiriz, 2018, Kumar and Patel, 2010]. However, we are interested in determining the clusters via each items' individual sales entries, not by aggregating sales. As mentioned in Section 4.2.1, we then used each cluster centroid as an additional model predictor. In practice, we used *k-means* clustering to determine similarities between the sales factor of each product.

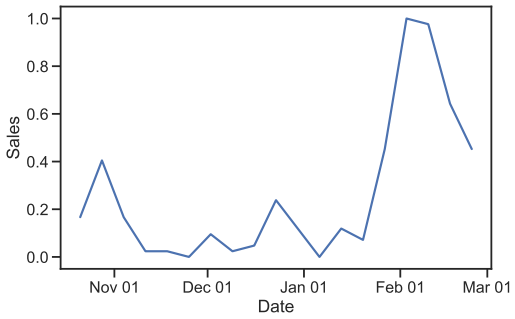
Instead of directly applying the clustering algorithm over all products, we approached the problem independently for each category. Moreover, we worked alongside retail managers to group *Seasons* according to their length and business goal, into multiple *Season Groups*. For instance, very short *Seasons* that target festivities, e.g., Easter and Father's Day, are grouped into the Festivities *Season Group*. We then ran the clustering algorithm independently for products within these groups. Table 4.2 lists the groupings used per each category.

Note that, as mentioned in Section 4.1.1, *Permanent* items have long historical data usually spanning more than a year, meaning the clustering approach is only applicable to *In and Out* and *One and Done* products. Table 4.2 summarizes the product distribution used for clustering per each *nature*. We can observe the clear differences in assortment, e.g., *Office* has no *One and Done* items while *Textile* has no *In and Out* products.

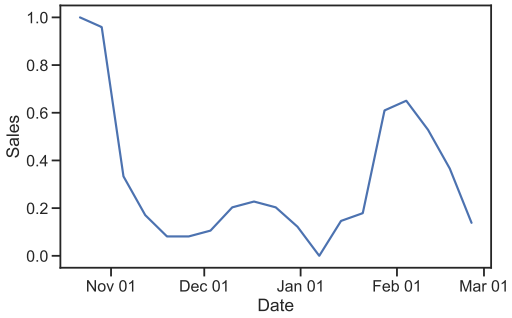
[3] With the exception of one experiment with data up to February, described in Chapter 5.



(a) Early performance peak



(b) Late performance peak



(c) Two performance peaks

Figure 4.5: Examples of common product life cycles observed for *One and Done* or *In an Out* products. Weekly sales values are normalized within the registered time window, i.e., the product's *Season*.

Table 4.2: Distribution of SKUs eligible for clustering, i.e., *In and Out* and *One and Done*.

(a) By nature

Category	**SKU Nature**	**# SKUs**
Home	In and Out	6
	One and Done	2037
Office	In and Out	5030
	One and Done	0
Textile	In and Out	0
	One and Done	33936

(b) By Season Group

Category	**Season Groups**	**# Seasons**	***k***
Home	"Home Special"	12	5
	Fall	12	4
	Winter	3	5
Office	Trimesters	16	3
	Festivities	4	3
Textile	Spring-Summer	5	5
	Fall-Winter	4	4

Additionally, we found that clustering directly with sales values was too sensitive to outlier weeks, i.e., weeks with sales spikes or downturns that are not representative of the product's natural life cycle. Instead, we opted to use a more stable measure, which we defined as the *sales factor* (λ_w^r). For week w and product r, λ_w^r corresponds to the ratio between the Moving Average (MA) of last 4 weeks of sales over the MA of the last year, if possible. It is calculated according to Equation 4.1:

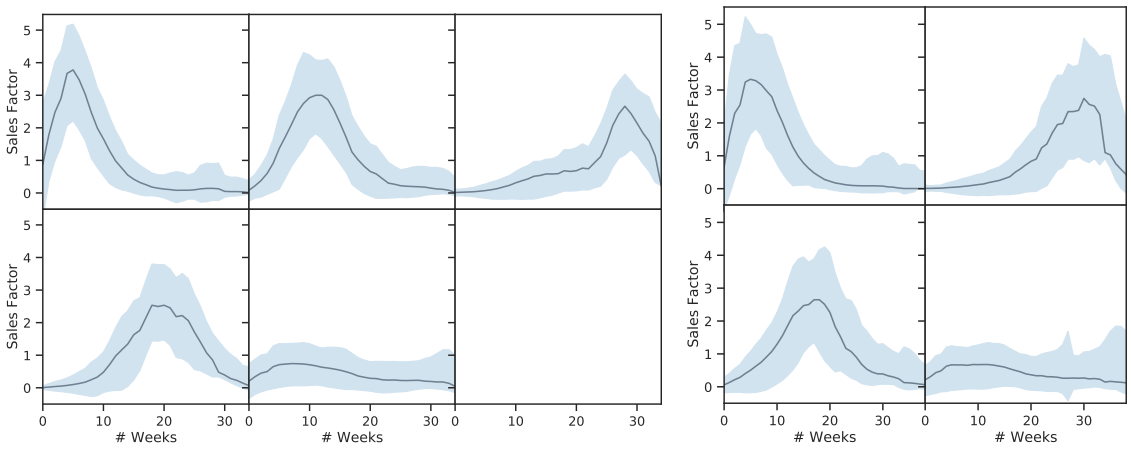
$$\lambda_w^r = \frac{\text{"}Sales\ MA\ last\ 4\ weeks\text{"}}{\text{"}Sales\ MA\ last\ 52\ weeks\text{"}} = \frac{\frac{1}{4}\sum_{i=0}^{3} s_{w-i}^r}{\frac{1}{n}\sum_{i=0}^{n} s_{w-i}^r} \qquad ,n = min(52, k_r) \tag{4.1}$$

s_w^r Sales at week w for product r
k_r Number of weeks product r has been on sale

Notice that λ_w^r is not sensitive to absolute sales values, only to shifts in a product's weekly sales. This allows products with different sales volume but with a similar life cycle pattern to be grouped together.

Applying *k-means* clustering requires k, i.e., the number of target clusters, to be specified. To do so, we executed multiple runs with k ranging from 2 to 12 within each Category x *Season Group* combination. We then selected the optimal k through the silhouette method [Rousseeuw, 1987], i.e., calculate the silhouette scores[4] for each run and select k which yields the highest average score. The resulting optimal k's are listed in Table 4.2.

Figure 4.6 shows the resulting clusters for the *Textile* category. We can observe each cluster is an accurate representation of the items it holds, represented by the blue shaded area. The different types of life cycles captured are noticeable: early, late, and even mid-season spikes. It also identified items with no distinguished life cycle. The resulting clusters for the remaining categories are displayed in Appendix A.



(a) Spring-Summer (b) Fall-Winter

Figure 4.6: Resulting clusters for each *Season Group* of the *Textile* category (dark line). The shaded areas illustrate the SKUs that belong to the clusters, represented by the standard deviation of the correspondent *sales factor* values.

[4] A measure of how similar an item is to its cluster compared to other clusters.

4.2.4 Computing Predictions

Given that we used weekly aggregated data to explore and train the models, we risked losing effects only captured via less aggregated data, e.g., at the store level. Even though predictions are generated at the SKU level across the retailer, we could not expect sales to be equally distributed across all stores. Naturally, some stores sell more than others, depending on, among others, their size and location.

Moreover, a product sale cannot occur if there is no stock. Stores may have large amounts of stocks for an item; others have no stock left; some are not selling it at all. By predicting aggregated sales without taking into account stock and sales distribution, we risk a positive prediction bias.

Considering this, we added an additional step to the forecasting procedure where we "split" sales across stores, given their weight on sales. For each SKU r at week w, the complete forecasting procedure is as follows:

1. The model is trained with SKU x Week data (*a priori*);
2. Let L be the set of the retailer's stores. For every product, we compute the sales factor (λ^r_w) for each store, i.e., $\lambda^r_{l,w}$, $l \in L$
3. Predictions are generated at the SKU x Week level, i.e., $\hat{s}^r_{w+i}$ for $i = 1,...,n$, where n is number of weeks until the end of the current *Season*, or up to a year for *Permanent* products;
4. Predictions are "split" across stores based on their λ^r_w, i.e., stores with higher λ^r_w get a higher share of the total weekly prediction, and vice-versa. We then have a "prediction" for each store ($s^r_{l,w}$);
5. For the following week ($w+1$), predictions are truncated given the item's available stock (I^r_w). This assumes there are on stock replenishments, which is applicable for *One and Done* and *In and Out* products. This and the following step are ignored for *Permanent* items;

$$\hat{s}^r_{l,w+1} = min(I^r_w,\ s^r_{l,w+1})$$

6. For the following weeks, the predictions are truncated given the previous weeks' prediction impact on stock.

$$\hat{s}^r_{l,w+i} = min(s^r_{l,w+i},\ I^r_w - \sum_{j=1}^{i} \hat{s}^r_{l,w+j}), \qquad i = 1,...,n$$

7. Lastly, the final predictions are aggregated back to the SKU x Week level:

$$\hat{s}^r_{w+i} = \sum_{l \in L} \hat{s}^r_{l,w+i}, \qquad i = 1,...,n$$

In practice, this process is executed as part of the data pipeline developed to integrate the predictions within the DSS. We take an in-depth look at the integration solution in Section 4.5.

4.2.5 Forecasting New Products

An additional challenge faced was how to approach new SKUs, which is a more difficult task because of the lack of direct historical product data. However, it was an essential effort to the work's success, as new items are introduced every *Season*, or even *mid-Season*.

In general, this is done via retailer managers' judgemental predictions, combined with market research. Within retailers who take a more analytical approach, the standard is to work via analogy. In practice, this involves forecasting new items by making use of past data of similar products [Fildes et al., 2019]. Conceptually, our approach was similar to that of Baardman et al., who use a joint approach of clustering and forecasting, and develop a classification model to assign new products to clusters [Baardman et al., 2018].

As described in Section 4.2.3, each of the developed clusters is a stable approximation of a given product's life cycle. In this sense, the intuition of our solution is to assign new products to one of the clusters, hence getting a medium-long perception of its life cycle. To achieve this, we developed a supervised classification model that predicts the probability with which a product belongs to a cluster, given the item's information (e.g., subcategory, base unit). We then used the cluster forecast as the single predictor for the SKU. As weeks go by, we get more data and fallback to the regular forecasting procedure.

Because when a new product is introduced a *Season* is already assigned to it (part of the managers' work), the cluster classification approach is done independently for each *Season Group*. This prevents making unreasonable classifications, e.g., a Christmas item being assigned to a cluster created from summer items.

More specifically, the solution is as follows, assuming the existence of the described classification model:

1. New SKUs are selected, i.e., items with less than four weeks of historical data. Let r be a new product;

2. A cluster is assigned to each product r, based on the classification model's predictions;

3. Sales forecasts are computed using the regular method (see Section 4.2.4) and only using the assigned cluster's centroid, i.e., $\hat{s}^r_{w+i}$ and $\hat{c}^r_{w+i}$, respectively;

4. For the product's first week of life, only the cluster's centroid is used to forecast the item's entire life cycle:
$$\hat{p}^r_{w+i} = \hat{c}^r_{w+i}, \qquad i = 1...,n$$

5. Let k be the number of weeks an r is in store. As weeks go by, there is more data and the regular forecasts are more accurate. Starting from the item's second week in store ($k = 2$), increasing weight is given to the regular forecasts. By the start of the fifth week in store ($k = 5$), it is no longer considered a new item. In this sense, at each week:
$$\hat{p}^r_{w+i} = \alpha * \hat{s}^r_{w+i} + \beta * \hat{c}^r_{w+i}, \qquad i = 1,...,n;\ k = 2,...,5; \qquad \alpha = \frac{(k-1)}{4},\ \beta = 1-\alpha$$

For the cluster prediction model, we formulated a multi-class classification problem. Naturally, there is no historical data for new items, meaning we could only have used product information as predictors. Similarly to the forecasting model, we experimented with multiple supervised learning algorithms: (i) Random Forests (RF); (ii) Extremely Randomized Trees (XRT); (iii) Gradient Boosting Machines (GBM). We then evaluated the results via 5 fold cross validation scores and selected the best performing model, ranked by resulting Logloss. We performed hyperparameter tuning as described in Section 4.2.2. The RF solution presented better overall scores and was the algorithm of choice. As in the forecasting model, we were limited to using product attributes are common across all categories, which reduced our options to product hierarchy information. Given the short amount of potential predictors, our feature engineering was minimal. Table 4.3 lists the final predictors used.

Table 4.3: Predictors used in the cluster classification model.

Predictor	Type	Description
Category	Categorical	The product's category, e.g., "Textile"
Subcategory	Categorical	Within the category, the product's subcategory, e.g., "Shorts"
Base Unit	Categorical	The product's base unit, e.g., "Men Summer Shorts"
Season	Categorical	The Season the product is listed in, e.g., "Summer"
Season Year	Numerical	The number of times the product's Season has existed.

4.3 PED Estimation Model Development

With the sales forecasting methodology underway, we then focused on the development of the Price Elasticity of Demand (PED) estimation model. As mentioned in Section 3.5, the solution's requirements related to price included: (i) "what-if" analysis of the impact of discrete price markdowns on demand and related metrics; (ii) optimal price markdown suggestion for underperforming products.

Considering this, the goal was to develop a supervised learning model able to predict the impact of applying a price markdown on a given SKU, hence giving and indirect measure of its estimated PED.

4.3.1 Overview

As mentioned in Chapter 2, the most common approach to determining how sales react to price is to model demand as a function of price (and other factors) via regression analysis, and use the regression coefficients to determine an item's PED (e.g., [Zaarour et al., 2016, Caro and Gallien, 2012, Smith, 2015]). Having a parametric formulation for demand allows to define the Markdown Optimization (MDO) problem via linear programming, such as the example formulation presented in Section 2.3.3. We studied the possibility of following the same techniques, but quickly realized

that the approach was unfeasible. In particular, using parametric regression techniques would hinder our sales forecasting procedure, which required a more sophisticated methodology, including the use of ML models, i.e., non-parametric regressors. Ferreira et al. reach similar conclusions, resorting to regression trees for modeling sales [Ferreira et al., 2016]. The downside is that we would have to follow a bespoke approach to MDO. We take an in-depth look at our MDO solution in Section 4.4.

Moreover, data exploration revealed that the effect that price has on an item's sales was reduced when considering its entire life cycle. The impact is more noticeable when looking at weeks directly after a price change. Figure 4.7 illustrates this by displaying a product's sales and price over its life cycle, highlighting where a price change occurred. We observed that the sales pattern varies significantly through time regardless of price. However, the price effect is pronounced directly after a price markdown, where we see a sales increase, which then fades out. We looked to model this local effect.

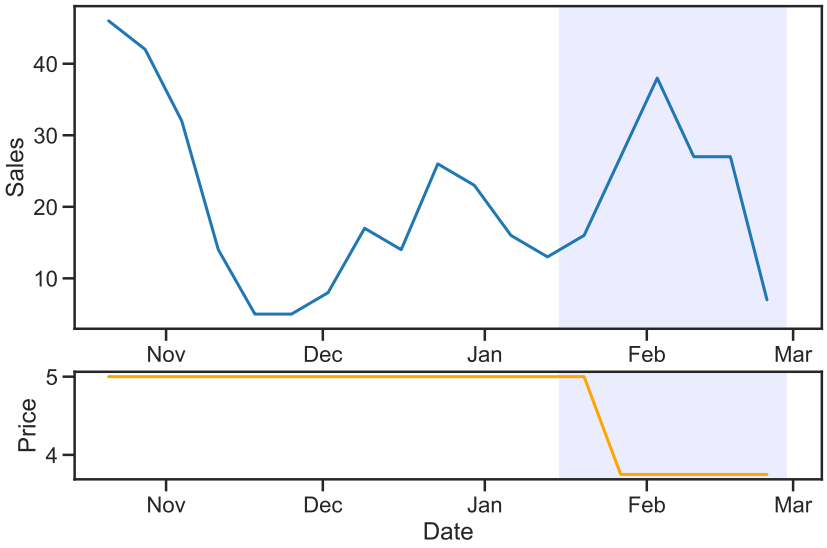


Figure 4.7: Evolution of an item's sales and price over its life cycle, highlighting the period where a price markdown occured.

Considering this, we computed the *sales variation* (Δ_w^r), i.e., the difference between sales two weeks after a price change and the sales of the two previous weeks. More formally, Δ_w^r is defined according to Equation 4.2.

$$\Delta_w^r = AvgSales_{+2\ weeks} - AvgSales_{-2\ weeks} = \frac{s_w^r + s_{w+1}^r}{2} - \frac{s_{w-1}^r + s_{w-2}^r}{2} \tag{4.2}$$

w Week where price markdown occured

s_w^r Sales at week w for product r

As previously illustrated in Section 3.5, the approach at estimating the effect that a price markdown has on sales was as follows:

1. From the entire data, weeks where price markdowns occurred were selected (e.g., 50% off), including the two previous and following weeks;

2. Weeks are labeled with the correspondent sales variation, i.e., Δ_w^r;

3. A supervised learning regression model was developed to predict Δ_w^r, given the price markdown, and additional predictors.

Considering this, the following subsections describe the steps taken to fulfill this solution.

4.3.2 Refining Data

As mentioned above, the first logical step was to isolate periods where price markdowns occurred. Some weeks overlapped with weeks with stock replenishments, leading to sales increases that most likely happened because of the inventory factor rather than the markdown. Because we wanted to single out the price effect (and not other factors), we added additional restrictions to reach the target data. The entire data selection process was as follows:

1. Isolate weeks where price markdowns occurred, computing the respective Δ_w^r;

2. Remove outlier cases, i.e., weeks with Δ_w^r above the 98th percentile, and below the 2nd percentile;

3. Remove weeks where major stock replenishments occurred, i.e., which we defined as an increase of 50% of an item's stock, in a given week. We chose not to remove all positive stock variation because product refunds translate into a stock increase;

4. Remove weeks with no stock, where sales cannot occur;

5. Remove weeks with a price reduction below 10%. These are most likely not real discounts, rather noise in the data, considering the approach taken to retrieve price information (see Section 4.1.2).

Similarly to the remaining predictive models, we performed the approach independently per category. The main finding was that the categories have different approaches to price markdowns, with *Office* having only applied price reductions to a small portion of its SKUs (30%). In contrast, *Textile* does so for nearly its entire collection (95%). Table 4.4 summarizes the data eligible for the PED estimation model.

Category	# SKUs w/ Markdown (% of total)	# Weeks	Avg Markdown per SKU (%)	Avg # Markdowns per SKU
Home	2457 (72%)	7775	30	3.16
Office	1671 (30%)	3540	32	2.12
Textile	32608 (95%)	109030	26	3.34

Table 4.4: Summary of eligible data after preparation for the PED estimation model.

4.3.3 Defining Predictors

Similarly to the sales forecasting solution (see Section 4.2.1), we needed to determine which predictors to include in the model. Our focus was two-fold: (i) to give preference to features related to price; (ii) to minimize the overlap of features with the sales forecasting model.

The primary analysis performed was assessing how Δ_w^r correlated with the applied markdowns. To our surprise, the main finding was that higher markdowns (50% to 75%) do not result in a higher sales increase. This may happen because markdowns this high are generally used to clear stock of less performant products. The price markdowns that yielded the highest sales variation are in the 25% to 50% range. Additionally, we observed that the response to price varies significantly across categories. Figure 4.8 illustrates the sales variation in response to common price markdowns.

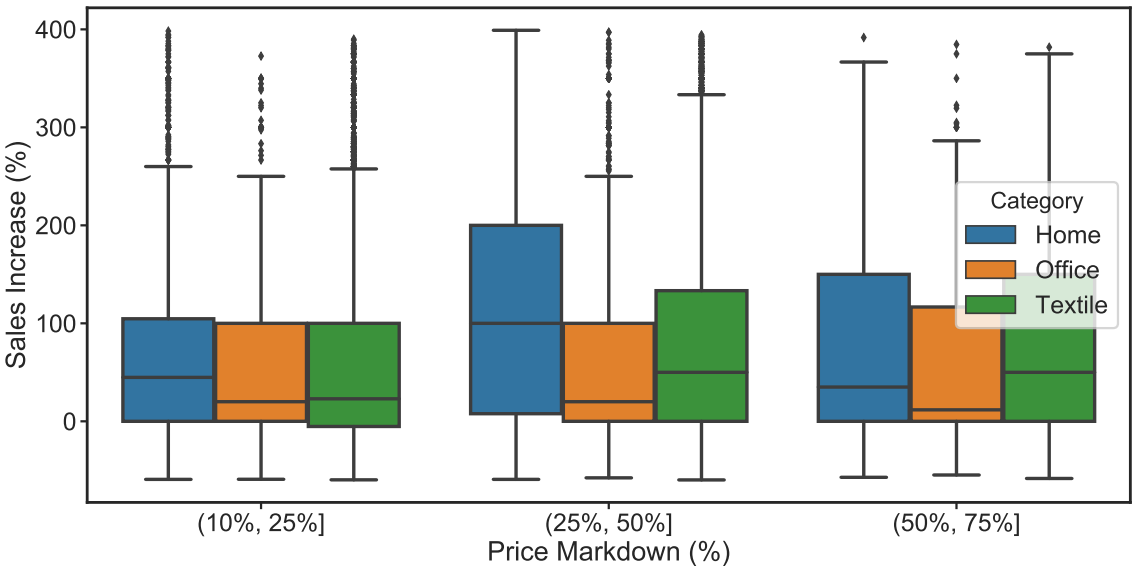


Figure 4.8: Distribution of the *sales variation* in response to the most common price markdowns, for the target categories i.e., *Home*, *Office*, and *Textile*. The *sales variation* is Δ_w^r, represented via percentage.

We extensively experimented with multiple feature combinations. We looked to add more informative price-related predictors (e.g., to account for the price of similar products), and information on the product hierarchy (e.g., to detect differences between subcategories). We now list the final predictors included in the model, with the respective context for the less obvious ones:

- **Price of Related Products:** as described in Section 2.2, the price of substitute or complementary products influences demand (cross-price elasticity of demand). We represented this as the percentile of the SKU's price among similar products, normalized between 0 and 1 (pricier is higher). Similar products were represented in two ways (hence, two predictors):
  1. Products within the same *Base Unit*, e.g., "men's summer shorts";
  2. Products within the same *Subcategory*, e.g., "shorts".
- **Applied Price Markdown:** the price markdown applied that week, represented via percentage (e.g., 25%);
- **Total Price Markdown:** the product's current discount compared to its original price. Cairo and Gallian suggest customers are more sensitive to the total relative markdown than to the absolute price cut [Caro and Gallien, 2012]. Indeed, retailers typically advertise the total discount applied to the product;
- **Previous Week's Performance:** Δ_w^r showed some degree of autocorrelation. We added the item's sales performance of the week before the markdown.
- **Average Response of Similar Products:** we computed the average *sales variation* to common price markdowns, i.e., 10%, 25%, 50%, 75%, for each group of similar products. Again, we represented similar products in two ways: (i) products within the same *Base Unit*; (ii) products within the same *Subcategory*.
- **Time Left in Store:** the number of weeks until the product's target outdate, i.e., the end of the current *Season*. Work reviewed suggested the impact of markdowns is reduced when reaching the outdate [Smith, 2015];
- **Stock:** the amount of stock available in-store. Despite the efforts to not include overlapping features with the sales forecasting model, the total stock amount represents a theoretical upper bound of the response variable.

The greatest effort towards this list was not deciding on the information to incorporate: it was clear that we should cover price and sales-related information. The main effort was on ensuring an accurate data representation, with a focus on preventing feature explosion and complexity. For instance, instead of representing sub-category and base unit information via dummy variables, we embedded this information via other numerical features (e.g., *Price of Related Products* and *Average Response of Similar Products*) [5].

4.3.4 Modeling Specifications

Facing a regression problem, it was crucial to experiment with multiple supervised learning algorithms and parameters. Initially, our approach was similar to the sales forecasting model (see Section 4.2.2), where we focused on using tree-based regressors, namely: (i) RF; (ii) XRT; (iii) GBM.

[5] This is akin to mean encoding.

Because the dataset is small for the *Office* and *Home* categories, (see Table 4.4), we opted to use all data for model training instead of leaving out a test set. However, our data is time-based, meaning we are subject to data leakage when evaluating model results via traditional cross validation with shuffling. Moreover, the target variable is representative of only an isolated period (two weeks), meaning time-series cross validation is not suited either [Hyndman and Athanasopoulos, 2018].

Considering this, we use *blocked cross-validation* [Bergmeir and Benítez, 2012], a variant of k-fold cross validation where each fold is split at a fixed timed interval, with a validation subsample. Figure 4.9 illustrates the procedure.

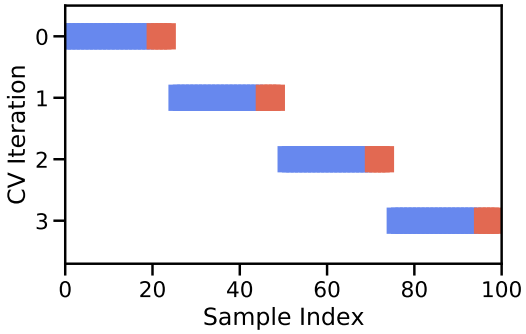


Figure 4.9: Illustration of *blocked cross-validation*. Blue and red periods represent training and validation set, respectively. Adapted from [Qamar-ud Din, 2019].

In particular, we split data into 5 equally sized folds. Moreover, to tune the models' hyperparameters, we used a Random Grid Search over the same values as the sales forecasting model, i.e., as specified in Listing 4.1. The first experiments following this framework led to the following conclusions:

(a) Training times were reduced, opening a path for more exhaustive model searches;

(b) The relationship between price related features and the response variable was not being captured, i.e., as price decreased, the *sales variation* did not necessarily increase.

Considering point (a), we looked for a more powerful modeling approach. We dropped the algorithms mentioned above in favor of H2O's AutoML, as initial tests yielded better results. As referenced in Section 2.4.4, AutoML is an approach at automating ML tasks, which includes training, tuning, and stacking multiple models in search of the best one. AutoML implementations usually share similar primary parameters to be defined, which under H2O's Python API include [H2O.ai, 2020]:

- *max_runtime_secs*: the maximum time that the AutoML process will run for, prior to training the final Stacked Ensemble models;
- *max_models:* the maximum number of models to build in an AutoML run, excluding the Stacked Ensemble models;

- *include_algos:* the list of algorithms to include during the model-building phase (e.g., XGBoost, GLMs, RFs).

For the latter challenge, i.e., (b), to ensure the relationship between price and sales followed intuition, we added a positive monotonic constraint between the feature *Applied Price Markdown (%)* and the response variable. Even though the aggregated data suggested this relationship might have different sensitivities depending on the category (see Fig. 4.8), the practical use of the model requires it to follow the Law of Demand: as price goes down, sales (may) go up [Marshall, 1898]. The major drawback was that it reduced our algorithm choice to boosting regressors, i.e., GBM and XGBoost [H2O.ai, 2020].

Considering this, Listing 4.2 includes the final AutoML configuration run[6].

```
automl_config = {
        'max_runtime_secs' : 3600,
        'max_models' : 50,
        'stopping_metric': 'mse',
        'sort_metric': 'mse',
        'include_algos' : 'GBM', 'XGBoost',
        'fold_columns': 'blocked_fold',
        'n_folds': 5,
        'monotone_constraint': {
            'price_markdown': +1,
        },
}
```

Listing 4.2: AutoML configuration specified.

4.4 Markdown Optimization Use Cases

Having finished the development of the first two components of the solution, i.e., the sales forecasting model and the PED estimation model, we now focused on making use of them to develop the practical use cases. As mentioned in Section 3.5, this included:

1. Identifying underperforming SKUs;
2. For underperforming SKUs, simulate the impact of multiple price markdown scenarios and compute the optimal pricing strategy.

4.4.1 Identifying Underperforming Products

Given the large number of SKUs each category comprises, the number of operational decisions to be taken at any point is overwhelming. Considering this, we looked to identify underperforming

[6] *blocked_fold* is computed previously for *blocked cross-validation*.

items, i.e., items that are not meeting stockout targets until the end of the *Season*, to bring managers' attention to them. Following this preventive reasoning enables setting price markdowns early, allowing for the possibility of revisiting the pricing decision later. Smith and Achabal claim this conservative mindset is preferable to setting late markdowns [Smith and Achabal, 1998].

Taking this into account, our solution was to use the sales forecasting model to compute predictions until the end of the *Season*. To this followed the calculating the expected amount of stock sold, comparing it to the retail managers' predefined targets. Figure 4.10 illustrates the situation for an item that did not meet the target, i.e., an underperforming product.

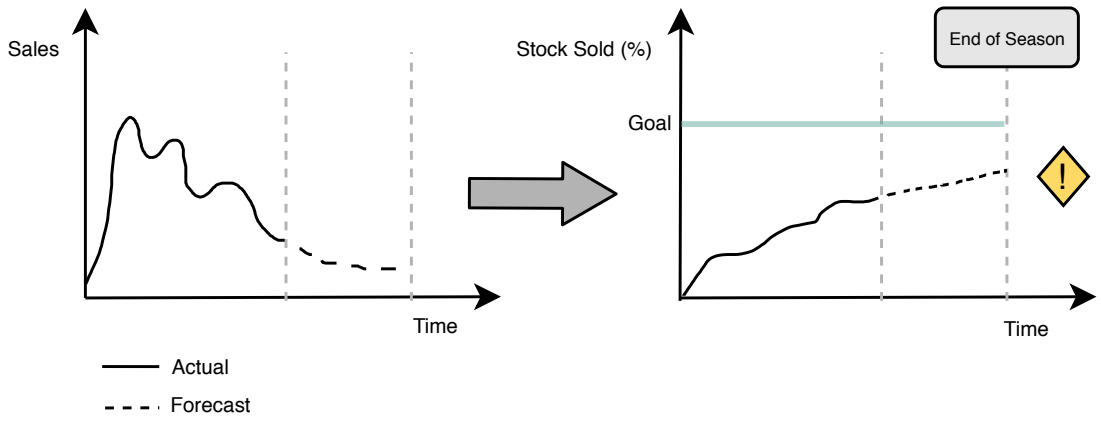


Figure 4.10: Illustration of the approach to detect underperforming products.

In particular, the approach is as follows:

Let I^r_{final} be the percentage of stock sold by the end of the current *Season*:

$$I^r_{final} = \frac{Units\ Sold}{Initial\ Stock} = \frac{\sum_{i=0}^{k} s^r_{w+i}}{I^r_0}$$

k	Number of weeks of the *Season*
s^r_w	Sales at week w for product r
I_0	Initial stock for product r

Each manager specifies I_{goal}, i.e., their target for the percentage of stock sold within the current *Season*. I_{goal}, typically ranges between 75% and 90%[7].

For each product, we compute the sales forecasts between the current week (w) and the last week the product is on sale ($w+n$). The sales forecasts are computed according to the method described in Section 4.2.4. We can then estimate $\hat{I}_{final}$, i.e., the forecasted stock sold:

[7]The target should be interpreted a lower bound for satisfactory performance, and thus is usually not set to 100%.

$$\hat{I}_{final} = \frac{Units\ Sold + Sales\ Forecast}{Initial\ Stock} = \frac{\sum_{i=0}^{k-n} s^r_{w+i} + \sum_{i=1}^{n} \hat{s}^r_{w+i}}{I^r_0}$$

n	Number of weeks left on the *Season*
$\hat{s}^r_w$	Sales forecast at week w for product r
I^r_0	Initial stock for product r

Finally, we signal out the SKUs where $\hat{I}_{final} < I_{goal}$, i.e., where the target is not met.

In practice, this process is executed as part of the data pipeline developed to integrate the use cases within the DSS. We take an in-depth look at the integration solution in Section 4.5.

4.4.2 Pricing Decisions

Having defined the methodology to identify underperforming products, we now focused on computing the respective optimal price markdowns.

The majority of the Markdown Optimization (MDO) works reviewed focused on maximizing directly for revenue (e.g., [Caro and Gallien, 2012, Zaarour et al., 2016, Alon et al., 2001]). However, the retailer's strategy favors selling products at a lower price if it ensures high stockout percentages. This reduces the costs associated with maintaining and transporting left-over stock, which in the case of *One and Done* items has no further use, and for *In and Out* means a long off-store time, hence, high storage costs.

Moreover, one of the requirements related to price was to enable "what-if" analysis over the impact of different price markdowns. This meant the solution should not output not only the optimal price scenario, but also enough information so that managers can evaluate the results of all scenarios and ultimately make a decision.

As like most retailers, price markdowns are defined from a set of predefined values, e.g., 25%, 50% off. However, the categories follow different strategies, each applying only their preferred discounts. Considering this, our pricing use cases are as previously illustrated in Figure 3.5. In particular, it takes two major steps, namely:

(a) **Price Markdown Simulation:** for underperforming products, we use the sales forecasts along with the PED estimations to compute the impact of different price markdowns on sales, by the end of the *Season*. Other relevant operational metrics are calculated, including price margin and stockout percentage. Only the standardly used markdowns are considered.

(b) **Price Markdown Suggestion:** we compute the optimal price markdown for the current scenario, considering the previously specified *stockout target*.

The reasoning behind (a) is that the sales forecasts establish a baseline, i.e., what to expect without changing the price, while the *sales variation* estimations complete them with *by how*

much sales are increasing if a price markdown is applied. Within scenarios that ensure the target is met, the one with the highest future cumulative margin is selected. In case no scenario estimates to reach the target, we optimize for future cumulative margin within all options.

More specifically, the computation of the simulated price markdowns is as follows:

Let M be the set of possible price markdowns, that is:

$$M = \{10\%, 25\%, 40\%, 50\%, 60\%, 75\%\}$$

For the current week, w, we compute the final sales forecast considering a price markdown m_i as:

$$ExpectedSales(m_i, w) = \hat{s}^r_w + \hat{\Delta}^r_w(m_i) \qquad m_i \in M$$

$\hat{s}^r_w$ Sales forecast (in number of units) for product r at week w
$\hat{\Delta}^r_w(m_i)$ Sales increase estimation (in number of units) given markdown m_i applied at week w

Which are computed until the end of the *Season*:

$$ExpectedSales(m_i, w, n) = \sum_{j=0}^{n} (\hat{s}^r_{w+j} + \hat{\Delta}^r_w(m_i)) \qquad m_i \in M$$

n Number of weeks left on the *Season*

Considering the updated sales forecast, we can compute multiple operational metrics, including the expected percentage of stock sold and the future total margin:

$$\hat{I}^r_{final}(m_i) = \frac{ExpectedSales(m_i, w, n)}{Initial\ Stock} = \frac{\sum_{j=0}^{n} (\hat{s}^r_{w+j} + \hat{\Delta}^r_w(m_i))}{I_0}$$

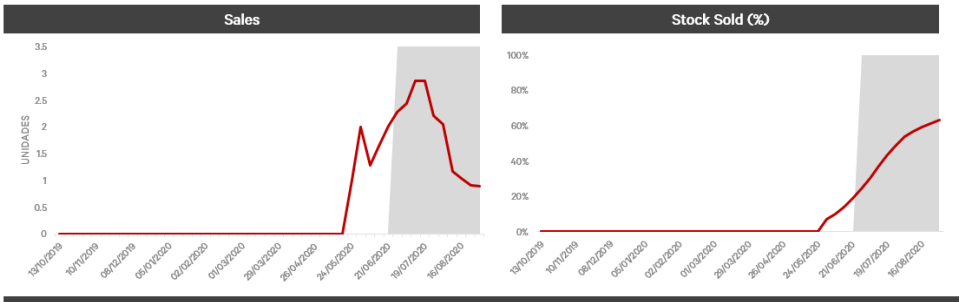
$$FutureMargin(m_i) = (m_i * P^r_0 - C^r) * ExpectedSales(m_i, w, n) \qquad m_i \in M$$

I^r_0 Initial stock for product r
P^r_0 Initial price for product r
C_r Purchase cost for product r

Considering that the eligible price markdowns are a small discrete set, it is feasible to compute the expected stock sold for all options, i.e., $\hat{I}^r_{final}(m) \ \ \forall m \in M$. We can then suggest the optimal price markdown.

These simulations are included within the DSS where managers observe the calculated metrics and the suggested price markdown, and ultimately make a decision, as illustrated in Figure 4.11. The markdown simulations are analyzed at the individual SKU level (Fig. 4.11a), where users can observe the forecasted sales and stock evolution. Furthermore, an individual analysis is presented for each price markdown, presenting relevant business metrics, including future units sold and

total stock sold. In this view, the suggested price markdown is highlighted. However, the intented use case is that price decisions be taken simultaneously for multiple SKUs to increase decision efficiency. This scenario is covered in a different view where only metrics related to the suggested price markdown are displayed (Fig. 4.11b).



Price Markdowns

Markdown (%)	Units Sold	Price	Unitary Margin	Future Margin	Total Margin	Stock Sold (%)
0%	24	17.99 €	8.68 €	205 €	231 €	63%
25%	28	13.49 €	5.02 €	142 €	168 €	75%
30%	29	12.59 €	4.29 €	126 €	153 €	77%
50%	**35**	**9.00 €**	**1.37 €**	**48 €**	**74 €**	**90%**
75%	39	4.50 €	-2.29 €	-89 €	-63 €	100%

(a) Individual product analysis with price markdown simulations.

SKU	Suggested Markdown (%)	Updated Price	Negative Margin ?	Total Margin (%)	EstimatedTotal Margin Variation	Estimated Stock Sold	Estimated Stock Sold Increase
7022937	50%	7.50 €	Não	35%	- 202 €	95%	25%
7054074	30%	5.59 €	Não	22%	- 121 €	79%	22%
7023331	25%	17.24 €	Não	41%	- 8 €	73%	6%
7030621	25%	7.49 €	Não	43%	30 €	100%	42%
6951779	30%	9.09 €	Não	37%	- 259 €	80%	13%
6988257	30%	6.99 €	Não	41%	- 52 €	45%	9%
6995890	50%	7.50 €	Não	44%	- 114 €	37%	9%
6951807	25%	12.75 €	Não	21%	- 228 €	96%	38%
7043420	50%	5.00 €	Não	65%	- 47 €	92%	5%
7054136	25%	5.99 €	Não	24%	- 250 €	100%	10%
6930718	50%	7.50 €	Não	42%	- 159 €	50%	17%
4249377	25%	3.75 €	Não	25%	- 5 €	96%	20%
5409236	30%	9.09 €	Não	36%	- 1 €	51%	22%
7030602	25%	5.25 €	Não	26%	- 150 €	100%	23%
7030630	50%	5.00 €	Não	24%	- 312 €	62%	14%
7054076	25%	5.99 €	Não	24%	- 135 €	100%	22%
7030626	50%	5.00 €	Não	24%	- 231 €	89%	43%

(b) Batch analysis with suggested price markdowns.

Figure 4.11: Price related use cases within the DSS.

4.5 Integration

The tail end of the methodology was to design and implement a deployment solution for data integration within the DSS. Moreover, the end use cases were tested and improved upon by running a pilot test, at the end of which a satisfaction survey was employed.

4.5.1 Architecture

As listed in Section 3.5.1, throughout the multiple stages of this work we used different systems depending on the task at hand. The execution of the solution relied heavily on the successful

integration of all components.

In this light, the architecture of the production system is as illustrated in Figure 4.12. Raw data is held in a private Apache Hadoop cluster, which is accessed via queries submitted to a connected Apache Impala engine. The main component of the solution is a Python codebase hosted in a Cloudera Data Science Workbench (CDSW) cluster which contains most of the functionality, including the data pipeline orchestration, and model training and scoring via an H2O server. Finally, the data is loaded to the DSS, which requests the processed data to the cluster.

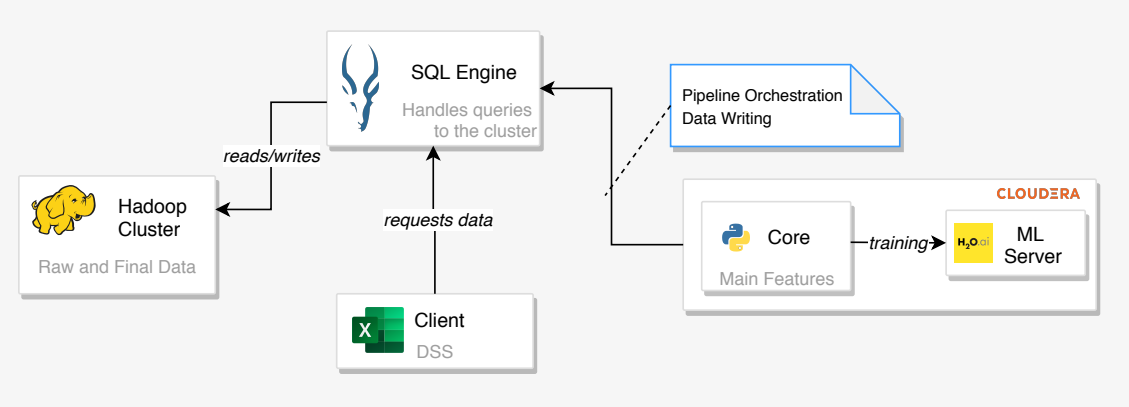


Figure 4.12: Components architecture of the production system.

4.5.2 Deployment Pipeline

The DSS is expected to update weekly given new available data. This involves obtaining raw data and performing all steps up to its final form. Additionally, the supervised learning models should be updated periodically to ensure quality results for recent data.

Moreover, daily data is available with a two-day delay, meaning that complete data of the previous week is only available each Monday afternoon. However, despite having a weekly data update, the DSS is meant to be used by Tuesday morning so that price changes and other operational decisions can be applied throughout the rest of the week. This puts pressure on the efficiency and reliability of the deployment solution, leaving around 16 hours for a complete update of all categories, including the ones not yet supported.

To ensure the process is executed seamlessly, we designed and developed a pipeline that automatically executes all the required steps, from data extraction to integration within the DSS. As illustrated in Figure 4.13, the pipeline stages are as follows:

1. **Data Processing:** we extract, transform, and load the raw data into new tables that follow the final data format. As mentioned in Section 4.1.1, this process is a set of SQL scripts submitted to the Apache Impala engine which are orchestrated by the core module. Additionally, the transformed data is loaded into the core module where it is further processed, namely to compute the necessary features for all supervised learning models;

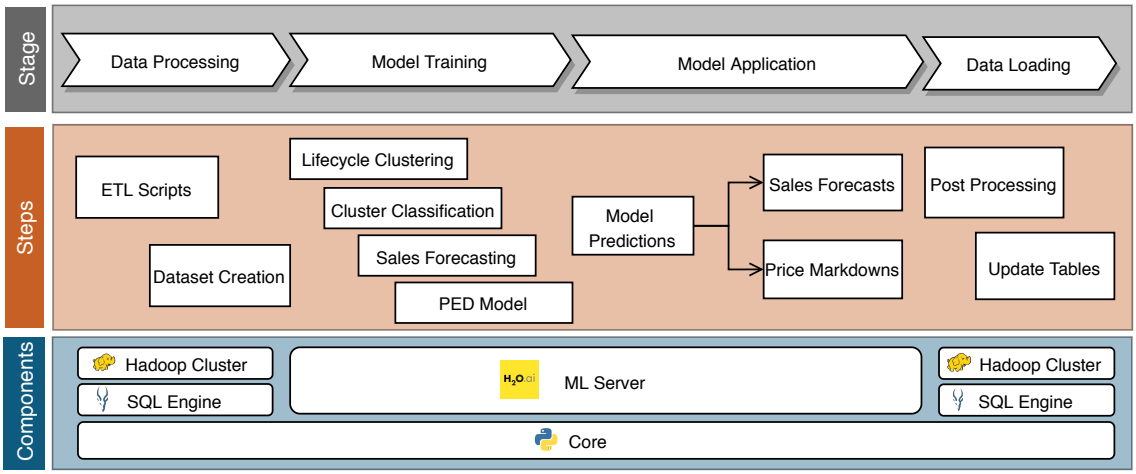


Figure 4.13: Illustration of the weekly deployment pipeline.

2. **Model Training:** the production supervised models are re-trained given new data. Since this is the most time-consuming stage and the benefits of a weekly model update would yield little impact, this process is scheduled monthly. The production models are stored to disk as binary files to be used for scoring during the *Model Application* phase;

3. **Model Application:** the production models are used to score the current week's data and to compute the use cases described thus far, i.e., the sales forecasts (including cluster assignment), price markdown simulation and suggestion;

4. **Data Loading:** the final data is loaded to a set of production tables from where the DSS requests data. Similarly to the *ETL Process*, this is a set of SQL scripts whose execution is orchestrated by the core module, and includes the integration of all DSS data, not just the use cases described in this dissertation.

4.5.3 Pilot Test

In the mid-stages of the development process, a pilot test was executed alongside the end users of the DSS, i.e., multiple managers of each category. Throughout four weeks, weekly meetings where held with representatives of each category with the primary goal of getting feedback on the developed solutions, namely the use cases supported by the supervised models, and the overall DSS experience.

One of the typical experiences held was to display sales forecasts and price markdown estimations for products whose behavior was familiar to the managers, to confirm if the right effects were being captured. This helped identify weak spots, edges cases, and bugs on the approach, which were then improved upon. For instance, initial iterations showed a tendency to underestimate demand during the spikes around October for *Home* products and Christmas for *Office* items, as pointed out during the held meetings. This motivated the implementation of the *Similar Product Performance* feature (see Section 4.2.1) based on seasonality decomposition.

Additionally, one of the goals of the pilot test was to accept and employ the suggested price markdowns on a set of products. We would then monitor the items' sales and compare them to the expected behavior. However, due to the COVID-19 pandemic outbreak, internal regulations were set up to prohibit price changes throughout multiple weeks, making it impossible to perform such analysis.

Nevertheless, by the end of the pilot test, a questionnaire was issued to get feedback on the entirety of the project. Some of the questions were directly related to the solutions developed as part of this dissertation, so we make use of those questions as part of the validation. We take an in-depth look at the questionnaire and its answers in Chapter 5.

4.6 Summary

Before implementing the components of our solution, we focused on cleaning, processing, and exploring the available data. We developed a procedure akin to an ETL process, which transformed the data into a workable format for further development, namely by aggregating daily operational data into weekly data. We also applied a *Seasons* system which helped in determining more accurately when products were in-store.

To implement the sales forecasting model, we experimented with multiple sales and seasonality related features, of which we listed the final set. To increase predictive capability, we extensively searched over hyperparameters options available for tree-based regressors. We devised an approach to identify life cycle patterns via clustering, whose centroids are used as an additional feature and to introduce stable predictions for new products, for which there is no historical data. We then computed weekly sales forecasts for all items and identified the ones underperforming by checking if target stockout targets are met by the end of the current *Season*.

To formulate the PED estimation model, we visually explored the relationship between price and sales and found the price effect to be noticeable in weeks close to price markdowns. We then used multiple tree-based algorithms to define a regressor that models the sales variation in response to a price markdown, considering the weeks directly before and after the change. We included multiple price and product related predictors, including the price of similar products. We experimented with multiple tree-based algorithms and AutoML in an attempt to achieve additional predictive power.

Furthermore, we compute price markdown scenarios by combining the effects of both models. In particular, we add to the original forecast the estimated sales variation of applying a given markdown, chosen from a set of typically employed discounts. These scenarios are then used to select the optimal price markdown, based on the expected stock sold at the end of the current *Season*, which has to surpass a predefined target. If no scenario reaches the target, we select the option that maximizes future total margin.

We then designed and implemented the deployment pipeline by leveraging the multiple infrastructure components used by the retailer, i.e., a Hadoop cluster for storage with Impala as SQL engine, and a CSDW cluster for development and production code. The core module is a Python

based app that orchestrates the pipeline steps, namely data extraction and processing, model training and application, and data deployment. This process is scheduled to run on a weekly basis.

Parallelly to the late stages of development, we launched the pilot test, which lasted for four weeks and included the collaboration of all managers using the DSS, who individually gave feedback during weekly meetings, which guided an iterative improvement of the solution. By the end of the pilot test, a satisfaction survey was introduced.

Chapter 5

Results

We now present the results of the work described thus far. Section 5.1 describes the modeling results of the sales forecasting solution. Likewise, Section 5.2 lists the results of the PED estimation model. Section 5.3 presents the analysis performed to evaluate stock sold estimation errors. Section 5.4 describes the results of the pilot test, and reviews relevant questions from the performed survey. Finally, Section 5.5 summarizes the main findings.

5.1 Sales Forecasting

We now describe the modeling results of the sales forecasting solution, which we validated by comparing weekly forecast errors with the approach currently in place by the retailer.

5.1.1 Metrics Used

In the context of forecasting, there is some debate in what metrics to use. We opt to use the Mean Absolute Percentage Error (MAPE) together with Bias. Percentage errors have the advantage of being unit-free, and so are used frequently to compare forecasts between datasets [Hyndman and Athanasopoulos, 2018], which is applicable to this work when comparing categories. Moreover, MAPE is commonly used in the context of retail forecasting because of its interpretability, which

is crucial when there is the need to explain forecasting accuracy to non-specialists [Fildes et al., 2019]. MAPE is expressed in the form of a percentage, according to Equation 5.1.

$$MAPE = \left(\frac{1}{n} \sum_{i}^{n} \frac{|Forecast - Actual|}{Actual} \right) * 100 \tag{5.1}$$

Bias, also referred to as Mean Percentage Error (MPE), differs to MAPE by considering the error instead of the absolute error in the denominator, providing a measure of a model's tendency to either under or over-estimate the target values [Swamidass, 2000]. Bias is also presented as a percentage, according to Equation 5.2.

$$Bias = \left(\frac{1}{n} \sum_{i}^{n} \frac{Forecast - Actual}{Actual} \right) * 100 \tag{5.2}$$

5.1.2 Baseline

Every forecasting or supervised learning task should have a baseline to improve upon. We used the forecasting approach currently in place by the retailer, i.e., projecting the average weekly sales of the SKU since the beginning of the *Season*. Using average values is a typical baseline in the context of forecasting, and in certain conditions yields surprisingly good results [Hyndman and Athanasopoulos, 2018].

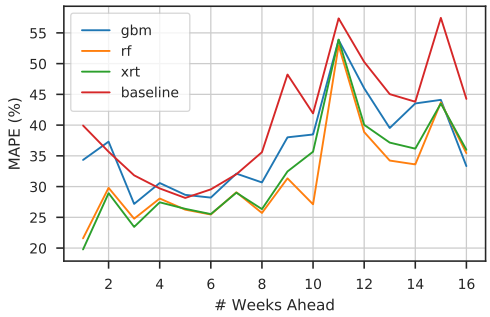
5.1.3 Modeling Results

As mentioned in Section 4.2.2, we tested results in data from the second half of the Spring-Summer *Season*, the largest of 2019 and the one with the most products in common. In all scenarios, we compared the best model of each algorithm, obtained by the procedure described in Section 4.2.2.

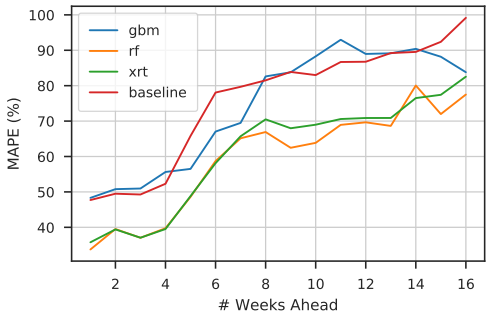
For *Permanent* products, we are interested in understanding our continuous forecasting capability, namely the tendency of our prediction to under or over estimate demand. Figure 5.1 displays the weekly MAPE of forecasting up to 16 weeks in advance, for all categories. Generally and as expected, the error tends to raise as the horizon stretches.

Table 5.1 summarizes the results, which further confirms that the developed approach unanimously outperforms the baseline, with a MAPE improvement of 8.93 percentage points (pp), 16.38pp, and 17.66pp, for *Home*, *Office*, and *Textile*, respectively. Furthermore, the most accurate algorithms (highlighted in bold) also reduce the bias in the *Home* (14.51% to 9.32%) and *Office* (28.24% to 22.37%) categories, while over-estimating for *Textile* products (-4.87% to 10.83%). In general, the algorithms performed similarly, with GBMs proving to be less biased, despite having worse accuracy in two of the three categories.

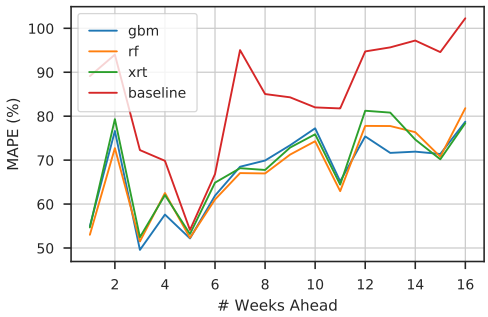
We performed the same analysis for *In and Out* and *One and Done* products, as illustrated in Figure 5.2. Similarly to the *Permanent* products, errors rise significantly as the horizon increases. All algorithms show an improvement over the baseline approach, with the RF and XRT appearing to have an overall better performance, specially in very short horizons (Figs. 5.2a and 5.2c).



(a) *Home*



(b) *Office*



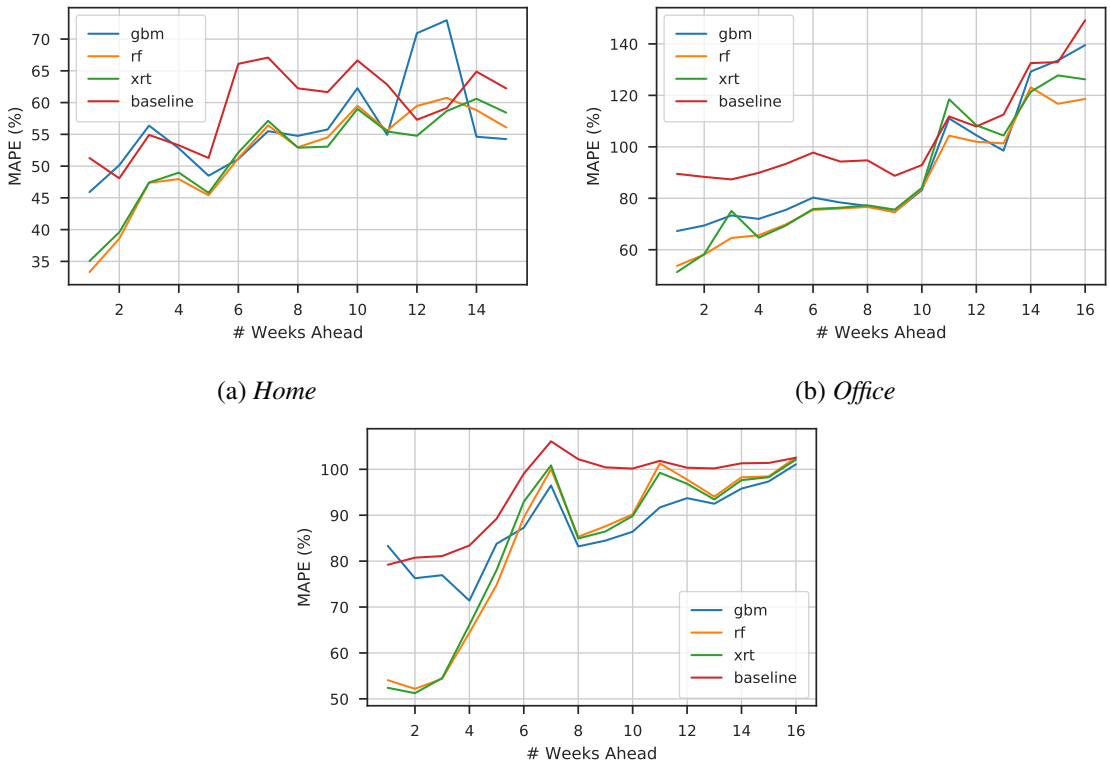
(c) *Textile*

Figure 5.1: Evolution of weekly MAPE results over for *Permanent* products when forecasting up to 16 weeks ahead.

Table 5.1: Summary of the forecasting results for *Permanent* products.

Category	Home		Office		Textile	
	MAPE (%)	Bias (%)	MAPE (%)	Bias (%)	MAPE (%)	Bias (%)
Baseline	40.67	14.51	75.90	28.24	84.92	-4.87
GBM	36.61	2.62	74.17	13.96	**67.26**	10.83
RF	**31.74**	9.32	**59.52**	22.37	67.50	13.33
XRT	32.60	8.44	61.28	21.92	68.80	13.32

The results for *In and Out* and *One and Done* products are summarized in Table 5.2. The best model of each category, i.e., with the minimum MAPE, presents an improvement over the baseline of 12.42pp, 18.63pp, and 11.70pp for *Home*, *Office*, and *Textile*, respectively. However, the best model does not always improve the forecast Bias, with *Home* showing a tendency to over-estimate demand more than the baseline (26.97% and 19.52%). For the remaining categories the Bias is reduced while showing a tendency to overestimate demand, i.e., for *Office* (-8.38% to 3.71%) and *Textile* (79.90% to 48.71%).



(a) *Home* (b) *Office*

(c) *Textile*

Figure 5.2: Evolution of weekly MAPE results over for *In and Out* and *One and Done* products when forecasting up to 16 weeks ahead, for each category.

Table 5.2: Summary of the forecasting results for *In and Out* and *One and Done* products.

Category	*Home*		*Office*		*Textile*	
	MAPE (%)	Bias (%)	MAPE (%)	Bias (%)	MAPE (%)	Bias (%)
Baseline	65.84	19.52	103.90	-8.38	95.75	79.90
GBM	56.05	24.87	91.71	-7.96	87.60	67.23
RF	51.86	24.06	**85.27**	3.71	**84.05**	48.71
XRT	**51.42**	26.97	88.40	6.17	84.06	49.54

5.2 PED Estimation

We now describe the results of the implementation of the PED estimation model, as described in Section 4.3.4.

5.2.1 Baseline

Before this dissertation's project, there was no standardized methodology for analyzing the impact of price on sales. Thus, there was no internal baseline to compare our approach to. We looked to define an intuitive and yet powerful baseline, considering both the retailer's product hierarchy and

information on typical price markdowns used. Considering this, we opted to project the historical average of *sales variation* in response to each common price markdown, i.e., (10%, 25%, 50%, and 75%), within each base *Base Unit*. In other words, the average *sales increase* of similar products, after a given markdown is applied.

5.2.2 Modeling Results

As mentioned in Section 4.3.4, we experimented with multiple supervised learning algorithms and validated results through blocked cross-validation. In parallel to the sales forecasting algorithm, we do not consider data from 2020. We ranked the models within each algorithm by Median Squared Error (MSE), and further compared the best model of each approach.

For the resulting models, we evaluated additional metrics, including the Median Absolute Error (MAE) within each typical price markdown, i.e., 10%, 25%, 50%, and 75%. Despite having introduced AutoML, we chose to evaluate the results of all algorithms experimented, including RF and XRT (which do not allow for monotonic constraints), to quantify the accuracy improvement (if any) of AutoML.

Table 5.3 summarizes the results. AutoML outperforms the alternatives, showing an MSE unanimously inferior in all categories. The same generally applies to the remaining metrics, except for the *Home* category, where the RF and XRT algorithms show an inferior Median error and MAE for the multiple price markdowns. Nevertheless, an inferior MSE indicates better overall performance, given its additional weight on larger errors. Considering the functional usage of the model, we aim to reduce the chance of over or under-estimating PED by a large degree, which would lead to financial losses.

Table 5.3: Summary of modeling results for the PED estimation model.

Category	Approach	R2	MAE	MAPE (%)	MSE	Median Error	MAE 10%	MAE 25%	MAE 50%	MAE 75%
Home	Baseline	0.036	83.06	61.70	353540.69	15.55	126.00	57.16	30.25	26.45
	GBM	0.44	47.66	43.27	202933.01	5.095	97.54	47.99	20.31	15.63
	RF	0.44	44.86	43.37	201920.17	5.25	77.51	50.49	20.00	10.88
	XRT	0.53	42.15	44.08	171932.21	5.50	81.27	45.19	16.54	16.81
	AutoML	0.59	53.73	62.74	**147278.95**	16.98	110.63	48.74	29.84	22.871
Office	Baseline	-1.90	15.64	73.93	7131.46	2.59	7.80	9.51	14.11	32.46
	GBM	0.50	8.31	51.94	1214.69	1.49	4.05	6.31	9.17	14.36
	RF	0.54	7.68	64.04	1117.84	1.87	3.41	5.96	8.84	11.55
	XRT	0.49	8.53	49.61	1356.64	1.46	4.19	5.73	9.68	17.08
	AutoML	0.56	7.63	49.21	**1080.13**	1.40	3.54	4.73	8.97	15.59
Textile	Baseline	0.12	2.44	60.01	19.30	1.44	2.32	2.26	2.18	1.98
	GBM	0.68	1.58	92.97	6.92	0.96	1.78	1.64	1.49	1.04
	RF	0.71	1.51	83.89	6.30	0.92	1.65	1.56	1.43	1.04
	XRT	0.69	1.51	84.78	6.64	0.93	1.75	1.63	1.49	1.09
	AutoML	0.78	1.35	81.04	**4.72**	0.85	1.52	1.40	1.28	0.92

5.3 Operational Evaluations

The modeling results described thus far are indicative of theoretical model performance but do not directly represent the applicability within their final use cases. We performed further analysis

from an operational standpoint to quantify the expected improvements in the supported use cases.

5.3.1 Stock Sold Estimations

As described in Section 3.5, the primary usage of the sales forecasting solution is to identify underperforming products as a means to prioritize price markdowns decisions. The success of this use case relies on the quality of the medium-long term stock sold estimations.

Taking this into account, we evaluated the weekly error of our stock sold estimations. In particular, we computed the real and estimated values of the percentage of stock sold value (e.g., 70% and 80%), followed by the percentual error, i.e., the absolute difference between them (e.g., 10pp). By performing this for multiple weeks, this measure quantifies the performance of the "underperforming product identification" use case. From an operational standpoint, it also hints at how close to the end-of-season one must be to trust the model.

We calculated the error for all products at each week, comparing the developed solution (*forecast*) with the baseline approach (*baseline*) (see Section 5.1.2). Similarly to the results presented in Section 5.1, we computed forecasts 16 weeks ahead, using the tail end of the Spring-Summer *Season*.

Figure 5.3 illustrates the results of this experiment for each category. It displays the weekly average error per product, where the shaded area represents the standard deviation of the error for the *forecast* approach. As expected, the errors generally increase as the horizon progresses, regardless of the method. Overall, our models outperform the baseline throughout all periods, excluding in weeks 13-16 of the *Textile* category (Fig. 5.3c). Coincidentally, the typical discount period of the *Season* starts at the 13th week, which influences the typical category's sales pattern, and is, therefore, harder to forecast. Moreover, it is the category that relies the most on price markdowns (see Section 4.4), which are especially enforced during that period.

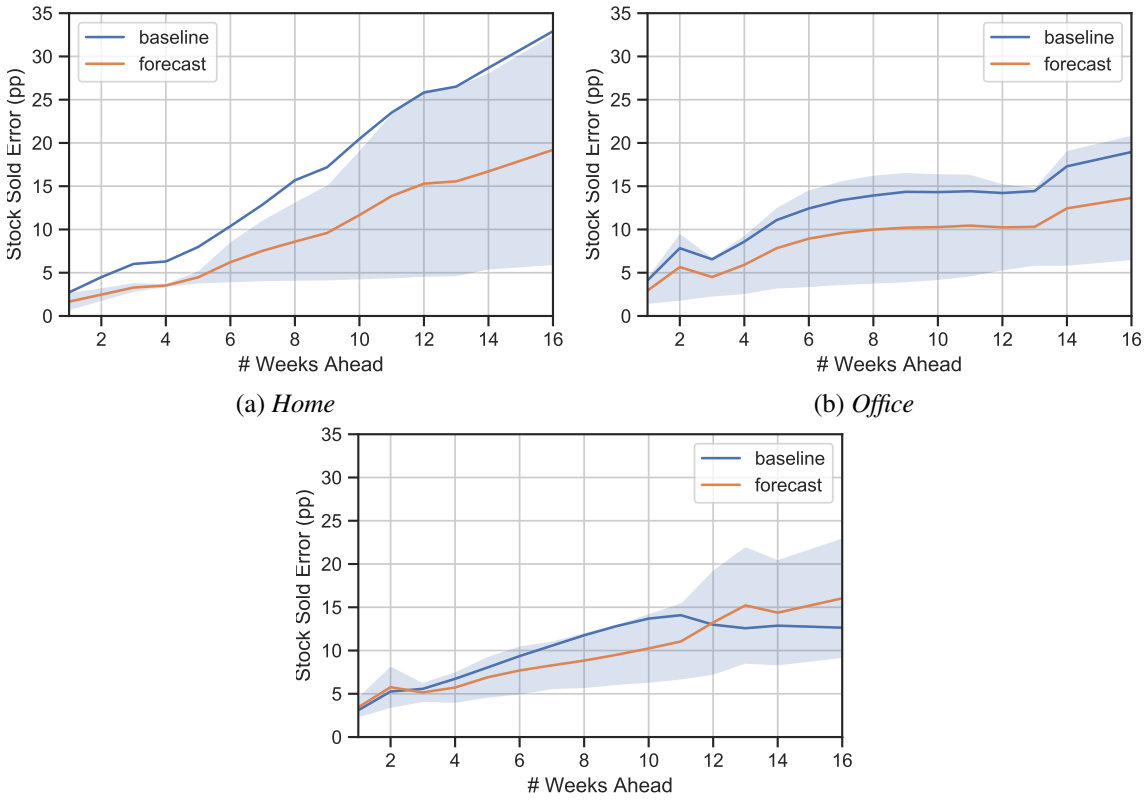
Nevertheless, the improvements are noticeable across all categories, as summarized in Table 5.4. In particular, we obtained an average weekly improvement of 7.17pp, 3.63pp, and 0.51pp, for *Home*, *Office*, and *Textile*, respectively, yielding an average stock sold estimation error of 9.59%.

Table 5.4: Summarized results of the stock sold estimation errors.

	Average Error (pp)		
	Baseline	Forecast	Difference
Home	17.02	9.85	-7.17pp
Office	12.75	9.12	-3.63pp
Textile	10.30	9.79	-0.51pp

5.3.2 Impact of the PED Estimation Model

Having confirmed the improvement in stock sold estimation error, we looked to quantify the impact of using the PED estimation model to correct forecasts, in situations where price markdowns



(a) *Home* (b) *Office*

(c) *Textile*

Figure 5.3: Weekly stock sold forecasting error, in percentual points.

are applied. To achieve this, we performed the same experiment just described in Section 5.3.1, but comparing the following approaches:

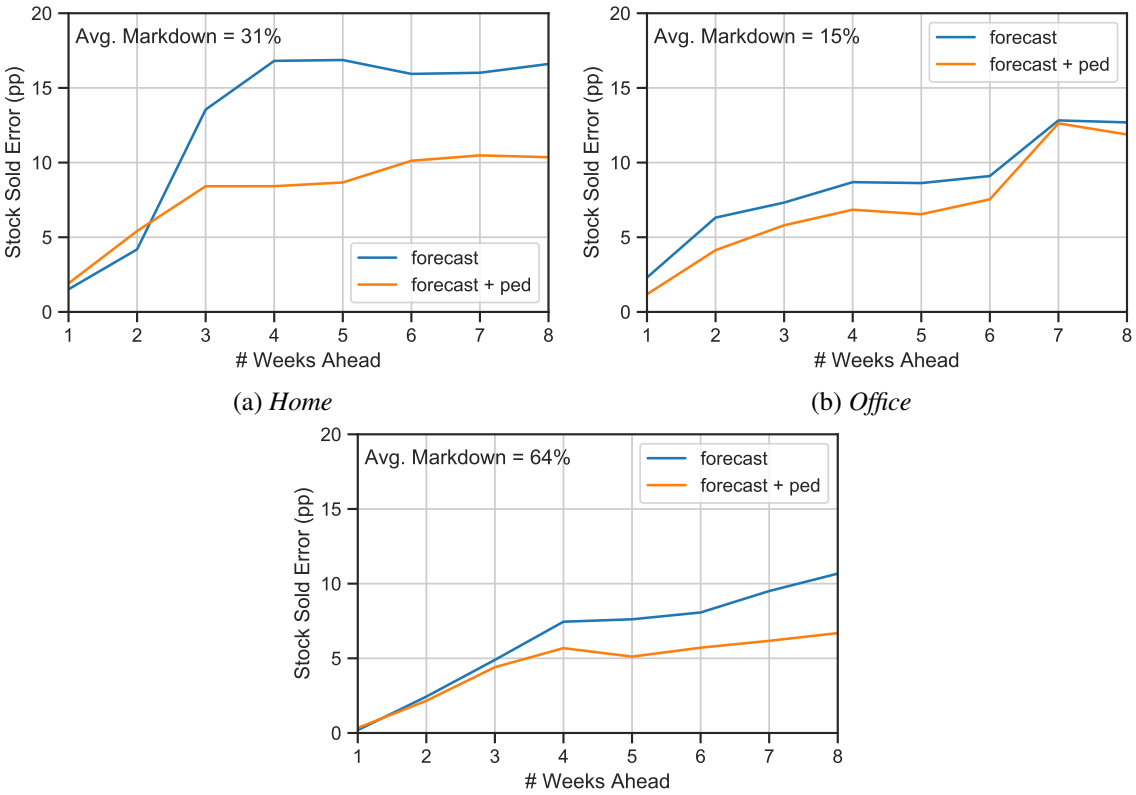
- Using the sales forecasting model (*forecast*);
- Using the sales forecasting model, adjusting forecasts according to the *sales increase* estimation, in weeks where a markdown is applied (*forecast* + *ped*).

Considering the goal of the experiment, we analyzed the results during a discount period, which typically lasts two months. In particular, we evaluate the most recent discount period available, i.e., between late December 2019 and February 2020[1].

Figure 5.4 display the results for each category. In general, we observed that the adjusted forecast succeeds in reducing the stock sold estimation error. However, the impact is noticeably different across categories. In specific, and as summarized in Table 5.5, the yielded improvements are of 4.71pp, 1.42pp, and 1.82pp, for *Home*, *Office*, and *Textile*, respectively.

Furthermore, the results are inconclusive in the relationship between the aggressiveness of the applied markdowns and the impact on model performance. *Office* (Fig. 5.4b) and *Textile* (Fig 5.4c) employed, respectively and on average, 15% and 64% markdowns, while yielding almost identical results.

[1] In specific, until the 23rd of February, before the COVID-19 pandemic outbreak in Portugal.



(a) *Home* (b) *Office*

(c) *Textile*

Figure 5.4: Weekly Stock Sold forecasting error, comparing *forecasting* and *forecasting + ped* approaches.

Table 5.5: Summarized results of the stock sold estimation errors, comparing the *forecast* and *forecast + ped* approaches.

	Average Error (pp)		
	Forecast	Forecast + PED	Difference
Home	12.68pp	7.97pp	-4.71pp
Office	8.48pp	7.06pp	-1.42pp
Textile	6.35pp	4.53pp	-1.82pp

5.4 Pilot Aftermath

The end of the pilot test described in Section 4.5.3 marked the formal ending of the project in which this dissertation is included. At this point, a six points Likert scale survey was employed to get feedback and quantify the overall satisfaction of the project as a whole. The target population were the retail managers that worked with the solution throughout the pilot test, totaling eight people. We evaluated the questions that were directly related to the work of this dissertation, as a form of additional validation. More specifically, the analyzed questions were as follows:

- **Q1:** In general, the solution met the expectations;

- **Q2:** The solution is useful for analyzing product performance and identifying underperforming items;
- **Q3:** The solution succeeds in responding to the problems it tackles, including price markdown planning and identifying underperforming items;
- **Q4:** The information presented in the "batch" view (Fig. 4.11b) is an efficient way to monitor price markdowns;
- **Q5:** The information presented in the "detailed" view (Fig. 4.11a) is an efficient way to monitor performance the details on price markdowns.
- **Q6:** The sales forecasting predictions presented in the DSS are reliable;
- **Q7:** The price markdown suggestions are adjusted to the items' expected sales pattern;

The results of the relevant portion of the survey are summarized in Figure 5.5. The results were substantially positive, with 56 answers lying on the positive side of the scale and the remaining 2 answers on the "Slightly Disagree" entry. The answers suggest the solution successfully fulfilled its intended use cases (Q1, Q2, and Q3). The most polarizing answers relate to *how* the use cases are presented (Q4, Q5), while still being generally accepted. Finally, the predictions generated by both models also received positive feedback (Q6, Q7), with a slight edge for the price markdown suggestions.

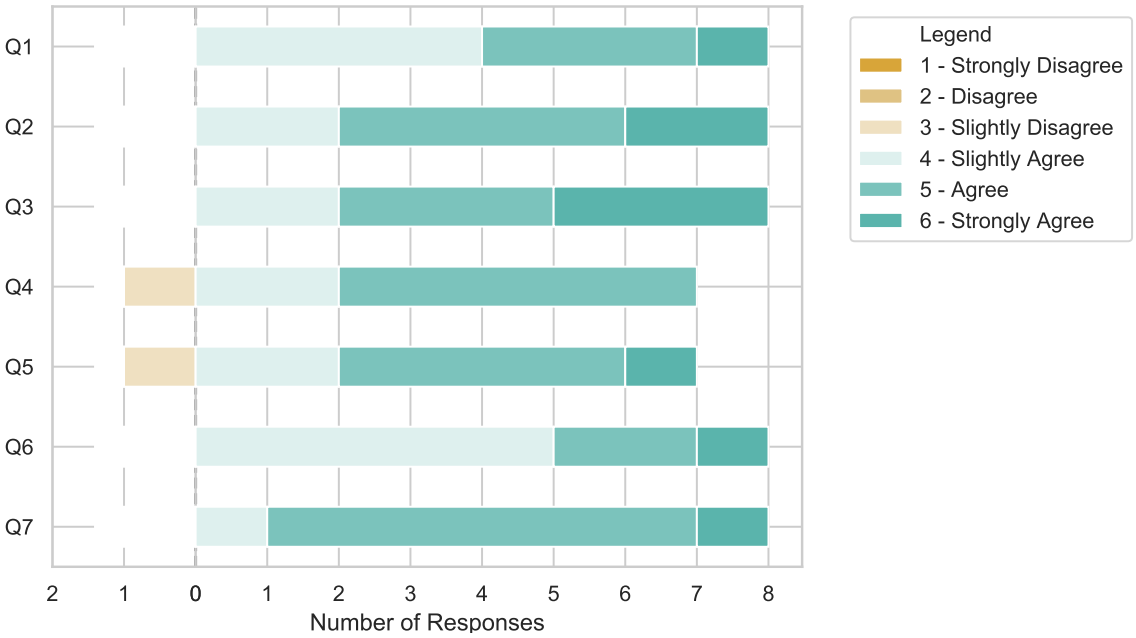


Figure 5.5: Summary of the responses to the relevant survey questions.

Given the pilot's success, the DSS is now part of the official workflow of the managers, after the weekly data update described in Section 4.5. Moreover, shortly after the termination of the

pilot test, the project was extended so that the approach is further applied to extra categories. As of June 2020, it is being deployed to an additional three categories, with plans to apply the approach to the entire division within a year. This further confirms the success of the developed methodology and respective implementation.

5.5 Summary

To evaluate predictive improvement, our solution's components were tested against the approaches currently in place by the retailer. From an operational and efficiency standpoint, we reviewed the results of the performed pilot test, including a review of the survey questions related to our use cases.

We evaluated the sales forecasting model performance during the second half of 2019's Spring-Summer *Season*, by computing weekly forecasts for up to 16 weeks ahead. As expected, the error increased as the horizon was further distant. No algorithm unanimously got better results. However, our solution consistently beat the baseline for all product natures, with an average improvement (measured in MAPE) of 14.32pp for *Permanent* items, and 14.91pp for *In and Out* and *One and Done* items, when considering the best model for each category.

Furthermore, we presented the modeling results for the PED estimation model. Given the lack of an analytical approach at pricing decisions, we established a baseline consisting of the historical average sales variation, in response to each standard price markdown, per *Base Unit*. The AutoML formulation consistently outperformed the other approaches, when ranking solutions via resulting MSE. However, the advantage was not unanimous when considering other metrics, e.g., median error. In any case, all algorithms significantly beat the baseline.

Additionally, we looked to measure performance through a metric directly related to the intended use cases. Considering this, for the same evaluation period, we computed the weekly stock sold estimation error. Once again, our solution consistently beat the baseline, showing an average improvement of 3.77pp across all categories. For discount periods, when adjusting the forecasts given PED estimations for weeks where price changes occurred, we yield an average 2.65pp improvement over just using the forecasting model.

Finally, the pilot test closure marked the end of the project under which this work arises. The survey results indicated the managers agree that the solution fulfills the ambitious goals initially defined. The DSS is now part of their weekly workflow for pricing decisions. Furthermore, the developed product is now being extended to additional categories, with plans to being applied to the entire division.

Chapter 6

Conclusions

In this chapter, we overview the work developed thus far. Section 6.1 summarizes the work done during this dissertation. Section 6.2 answers the research questions proposed. Section 6.3 lists our main contributions. Section 6.4 describes future work regarding the developed methodologies and respective project.

6.1 Summary

This work arises in the context of a project with a large Portuguese retailer with the primary goal of improving their product life cycle management and price markdown strategy through a computerized approach. Product life cycle (PLC) describes the typical pattern that products follow during their life, which is traditionally defined as having four stages: (i) introduction; (ii) growth; (iii) maturity; (iv) decline. Knowing at which stage a given product is in presents valuable information for all levels of the operational chain. However, in the context of retail sales and its environment, characterized by the evergrowing competitiveness and increased customer expectations thanks to the ubiquity of e-commerce, the traditional product life cycle is rarely followed. Instead, products have short and erratic life cycles, making operational decisions a hard task.

Despite this unpredictable setting, purchases from suppliers are typically made with months in advance, which may lead to less-than-expected sales and unsold stock. Retailers typically employ price markdowns, i.e., permanent price reductions, in order to stimulate sales and reduce costs. However, knowing when and by how much to apply markdowns is yet another challenging task - do it too soon and revenue is lost, do it too late and sales targets are not met. That is assuming a price markdown will affect sales as intended, which is not always the case.

Price Elasticity of Demand (PED) measures the relationship that a given price change will have on an item's demand. The value of this measure has been empirically calculated for only a handful of universal items, such as oil and water. For all others, estimations must be made in order to make informed pricing decisions. However, in traditional retail sales settings, including the retailer target of this work, price markdowns are applied following a judgemental decision based on managers' experience. Considering the large number of products for which operational decisions must be made at any point, this is, at best, a questionable approach.

In this context, this dissertation's work focused on improving the retailer's price markdown decisions. In particular, we targeted three categories of the retailer's *Non-food* division, each comprising thousands of products of a radically different type, behavior, and sales volume. To achieve this, we designed a three-fold approach.

First, we developed a sales forecasting model that predicts weekly sales for each product across all stores, with the primary objective of identifying underperforming products. We used supervised learning algorithms with multiple attributes known to influence demand. However, considering the short life cycle of most products, the amount of historical data per each item is reduced. To tackle this, we performed a clustering procedure on products' sales profiles to detect common patterns, and use each centroid as an additional predictor. Furthermore, we used the clusters to introduce stable forecasts for new items. The approach yielded an average improvement of 14.25pp in weekly sales forecasting error (measured in MAPE), and a 3.77pp improvement in stock sold estimation error.

The second component was a PED estimation model with the primary goal of predicting the demand increase of a product in response to a given price markdown. The approach was motivated by observing that the impact of price markdowns are most noticeable in the weeks directly after the change. Considering this, we isolated weeks where price markdowns occurred and used supervised learning algorithms to create a regression model that predicts the expected sales increase given a price change. By using these estimates in weeks where price changes occurred, we obtained an average improvement of 3.77pp in stock sold estimation error.

Ultimately, we combined the models to compute price markdown simulations and the optimal price markdown, following a strategy catered to the retailer's policy on stock management. In particular, it looked to ensure a predefined stock sold targets are met by the end of *Season*, while maximizing price margin. These use cases were integrated within a Decision Support (DSS) that is updated weekly, for which we developed an automated data pipeline that executes the required steps, including data extraction and preparation, model training, and deployment.

To validate the solution, namely the use cases supported by this dissertation's work, we ran a pilot test where weekly meetings were held for four weeks alongside the target end users of the DSS, i.e., category managers. Products can now be monitored, namely to detect underperforming ones so that price markdowns can be applied. The survey employed by the end of the pilot test indicated the solution was unanimously accepted. As an additional indicator of success, the project was extended to progressively integrate additional categories, up to the retailer's entire division.

6.2 Research Questions Revisited

Taking into account the results presented in Chapter 5 and the implementation steps taken towards this high-level approach at price decision improvement, we now revisit our research questions:

- **RQ1.** *Can the retailer's approach at sales forecasting be improved through an approach based on product life cycle pattern detection?* The forecasting solution unanimously outperformed the baseline approach, with the categories showing an average improvement of 14.23pp for *Permanent* products, and 14.25pp for *In and Out* and *One and Done* products. The forecasts yield an average stock sold estimation error improvement of 3.77pp, at 9.58pp of absolute error.

- **RQ2.** *Does updating a sales forecast through a price elasticity of demand estimation model provide an accurate simulation of arbitrary price markdown scenarios?*. Correcting the original forecasts with the developed PED estimation approach yielded a stock sold estimation error improvement of 2.65pp when considering discount periods. Furthermore, the results of the survey indicated that the presented simulations meet the managers' expectations and are suitable for practical usage.

- **RQ3.** *Does providing the retailer with the set of underperforming products and price markdown scenarios increase pricing quality and efficiency?* Identifying underperforming products enabled us to steer pricing decisions towards the most crucial cases. Unfortunately, monitoring the effect of price changes for comparison with the suggested scenario was not possible, namely due to restrictions imposed during the COVID-19 pandemic.

Overall, we are confident that the developed solution meets the proposed goals, with enough robustness for an extension to the retailer's entire division in the near future.

6.3 Main Contributions

As a result of this dissertation, our main contributions are:

Literature Review: we studied the relevant areas in taking informed decisions for price markdowns in the context of retail sales, namely methodologies and case studies on estimating PED, and Markdown Optimization.

Price Markdown Decisions: we designed a three-fold solution to support price markdown decisions. It combined a sales forecasting and a PED estimation model to compute price markdown simulations and respective optimal price markdown, relevant in scenarios where the priority is to ensure minimum left-over stock.

Empirical Validation: the designed solution was validated in a real-life scenario through a successful implementation and deployment in a large Portuguese retailer.

6.4 Future Work

Taking into account the results of the developed solution, the feedback of the pilot test's participants, and the inevitable limitations of this work, we present the following possible paths:

Model predictors: the predictors used throughout the supervised learning models were limited to information available on all target categories, given the need for generalization. Additional predictive power may lie in data only existing in certain categories. Extending the methodology to identify and use category-specific attributes automatically would be a natural area of improvement.

Markdown Optimization: the computed and suggested price markdowns assume that only a single markdown could be applied. This is a reasonable assumption considering the short time-window during which markdowns are applied. However, the pricing solution may benefit from an extension in which multi-step price markdowns are considered.

Combining data: the methodology proposed is followed independently per each target category. This was necessary considering the significantly different types of products, sales volume, and sales patterns of the target categories. However, when extending the solution to more categories, it may be beneficial to introduce a methodology to define *meta-categories*, i.e., a group of similar categories that benefit if approached in conjunction.

Bibliography

[Agarwal, 2014] Agarwal, S. (2014). Data Mining: Data Mining Concepts and Techniques. In *2013 International Conference on Machine Intelligence and Research Advancement*, pages 203–207. IEEE. Cited on pages 21 and 24.

[Ailawadi et al., 2006] Ailawadi, K. L., Harlam, B. A., César, J., and Trounce, D. (2006). Promotion Profitability for a Retailer: The Role of Promotion, Brand, Category, and Store Characteristics Promotional. *Journal of Marketing Research*, 2437(43):518–535. Cited on page 42.

[Alon et al., 2001] Alon, I., Qi, M., and Sadowski, R. J. (2001). Forecasting aggregate retail sales: A comparison of artifcial neural networks and traditional methods. *Journal of Retailing and Consumer Services*. Cited on pages 8 and 57.

[Baardman et al., 2018] Baardman, L., Levin, I., Perakis, G., and Singhvi, D. (2018). Leveraging Comparables for New Product Sales Forecasting. *Production and Operations Management*, 27(12):2340–2343. Cited on page 48.

[Babar et al., 2015] Babar, M., Nguyen, P. H., Cuk, V., and Kamphuis, I. G. (2015). The development of demand elasticity model for demand response in the retail market environment. *2015 IEEE Eindhoven PowerTech, PowerTech 2015*, pages 1–6. Cited on page 13.

[Barbosa, 2018] Barbosa, P. (2018). A machine learning approach to promotional sales forecasting. Master's thesis, University of Porto. Cited on pages 10, 27, and 28.

[Bergmeir and Benítez, 2012] Bergmeir, C. and Benítez, J. M. (2012). On the use of cross-validation for time series predictor evaluation. *Information Sciences*, 191:192–213. Cited on page 54.

[Bergmeir et al., 2018] Bergmeir, C., Hyndman, R. J., and Koo, B. (2018). A note on the validity of cross-validation for evaluating autoregressive time series prediction. *Computational Statistics and Data Analysis*. Cited on page 43.

[Boada-Collado and Martínez-de Albéniz, 2020] Boada-Collado, P. and Martínez-de Albéniz, V. (2020). Estimating and Optimizing the Impact of Inventory on Consumer Choices in a Fashion Retail Setting. *Manufacturing & Service Operations Management*, 22(3):582–597. Cited on page 10.

[Bouzdine-Chameeva and Hill, 2015] Bouzdine-Chameeva, T. and Hill, A. V. (2015). Pricing Strategy in a Clearance Period. In *New Meanings for Marketing in a New Millennium. Developments in Marketing Science: Proceedings of the Academy of Marketing Science*, pages 89–93. Springer, Cham. Cited on pages 13 and 14.

[Breiman, 1996] Breiman, L. (1996). Stacked regressions. *Machine Learning*. Cited on page 25.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine Learning*. Cited on page 23.

[Brownlee, 2016] Brownlee, J. (2016). *Master Machine Learning Algorithms Discover How They Work and Implement Them From Scratch*, volume 1. Jason Brownlee. Cited on pages 22 and 23.

[Carbaugh, 2010] Carbaugh, R. J. (2010). *Contemporary Economics: An Applications Approach*. Thomson/South-Western. Cited on page 12.

[Caro and Gallien, 2012] Caro, F. and Gallien, J. (2012). Clearance pricing optimization for a fast-fashion retailer. *Operations Research*, 60(6):1404–1422. Cited on pages 9, 10, 13, 16, 41, 49, 53, and 57.

[Chase, 2013] Chase, C. W. (2013). *A Structured Approach to Forecasting*. John Wiley & Sons, Inc. Cited on page 6.

[Chen and Simchi-Levi, 2012] Chen, X. and Simchi-Levi, D. (2012). Pricing and Inventory Management. In *The Oxford Handbook of Pricing Management*. Oxford University Press. Cited on pages 2 and 15.

[Coşgun et al., 2013] Coşgun, Ö., Kula, U., and Kahraman, C. (2013). Markdown Optimization via Approximate Dynamic Programming. *International Journal of Computational Intelligence Systems*, 6(1):64–78. Cited on page 2.

[Demiriz, 2018] Demiriz, A. (2018). *A Data Mining-Based Framework for Multi-item Markdown Optimization*. Springer Singapore. Cited on page 44.

[Dubé, 2004] Dubé, J. P. (2004). Multiple Discreteness and Product Differentiation: Demand for Carbonated Soft Drinks. *Marketing Science*. Cited on page 9.

[Dunne et al., 2013] Dunne, P. M., Lusch, R. F., and Carver, J. R. (2013). *Retailing*. Cengage Learning. Cited on page 2.

[Edgeworth and Marshall, 1891] Edgeworth, F. Y. and Marshall, A. (1891). Principles of Economics. *The Economic Journal*. Cited on page 10.

[Elmaghraby and Keskinocak, 2003] Elmaghraby, W. and Keskinocak, P. (2003). Dynamic pricing in the presence of inventory considerations: Research overview, current practices, and future directions. *Management Science*. Cited on page 16.

[Fernandes, 2020] Fernandes, N. (2020). Economic Effects of Coronavirus Outbreak (COVID-19) on the World Economy. *SSRN Electronic Journal*. Cited on page 44.

[Ferreira et al., 2016] Ferreira, K. J., Lee, B. H. A., and Simchi-Levi, D. (2016). Analytics for an online retailer: Demand forecasting and price optimization. *Manufacturing and Service Operations Management*, 18(1):69–88. Cited on pages 13, 14, 42, and 50.

[Feurer et al., 2015] Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., and Hutter, F. (2015). Efficient and Robust Automated Machine Learning. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 2962–2970. Curran Associates, Inc. Cited on page 25.

[Fildes et al., 2019] Fildes, R., Ma, S., and Kolassa, S. (2019). Retail forecasting: Research and practice. *International Journal of Forecasting*. Cited on pages 6, 8, 9, 48, and 66.

[Friedman et al., 2008] Friedman, J., Hastie, T., and Tibshirani, R. (2008). *The Elements of Statistical Learning*. Hastie, Tibshirani & Friedman, 2 edition. Cited on pages 23 and 24.

[Friedman, 2001] Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*. Cited on page 24.

[Friedman, 2002] Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 38(4):367–378. Cited on page 43.

[Garbade, 2018] Garbade, M. J. (2018). Regression Versus Classification Machine Learning: What's the Difference? Cited on page 21.

[Gijsbers et al., 2019] Gijsbers, P., LeDell, E., Poirier, S., Thomas, J., Bischl, B., and Vanschoren, J. (2019). An Open Source AutoML Benchmark. *arXiv preprint arXiv:1907.00909 [cs.LG]*. Cited on page 25.

[Gill, 2000] Gill, J. (2000). *Generalized Linear Models: A Unified Approach*. Sage Publications, Inc. Cited on page 7.

[Gilliland et al., 2016] Gilliland, M., Tashman, L., and Sglavo, U. (2016). *Business Forecasting: Practical Problems and Solutions*. Wiley. Cited on page 6.

[Gupta et al., 2006] Gupta, D., Hill, A. V., and Bouzdine-Chameeva, T. (2006). A pricing model for clearing end-of-season retail inventory. *European Journal of Operational Research*. Cited on page 15.

[H2O.ai, 2017] H2O.ai (2017). *H2O AutoML*. Cited on page 25.

[H2O.ai, 2020] H2O.ai (2020). *Python Interface for H2O*. Cited on pages 43, 54, and 55.

[Harsha et al., 2019] Harsha, P., Subramanian, S., and Uichanco, J. (2019). Dynamic pricing of omnichannel inventories. *Manufacturing and Service Operations Management*. Cited on pages 18 and 19.

[Hormby et al., 2010] Hormby, S., Morrison, J., Dave, P., Meyers, M., and Tenca, T. (2010). Marriott international increases revenue by implementing a Group Pricing Optimizer. *Interfaces*. Cited on page 13.

[Hu et al., 2016] Hu, M., Li, W., Li, L., Houston, D., and Wu, J. (2016). Random forests classification. Cited on page 24.

[Hutter et al., 2019] Hutter, F., Kotthoff, L., and Vanschoren, J., editors (2019). *Automated Machine Learning*. Springer. Cited on page 25.

[Hyndman and Athanasopoulos, 2018] Hyndman, R. J. and Athanasopoulos, G. (2018). Forecasting: Principles and Practice. *Principles of Optimal Design*. Cited on pages 6, 7, 41, 42, 43, 54, 65, and 66.

[Ingenbleek et al., 2003] Ingenbleek, P., Debruyne, M., Frambach, R. T., and Verhallen, T. M. (2003). Successful new product pricing practices: A contingency approach. *Marketing Letters*. Cited on page 15.

[Johnson and Bradford, 2002] Johnson and Bradford, C. (2002). Retail: The Wal-Mart Effect. *The McKinsey Quarterly*. Cited on page 6.

[Kang and Gershwin, 2005] Kang, Y. and Gershwin, S. B. (2005). Information inaccuracy in inventory systems: Stock loss and stockout. *IIE Transactions (Institute of Industrial Engineers)*. Cited on page 36.

[Kelleher and Tierney, 2018] Kelleher, J. D. and Tierney, B. (2018). *Data Science*. The MIT Press. Cited on pages 19, 20, 21, 23, and 42.

[Khouja, 2016] Khouja, M. (2016). Late season low inventory assortment effects in the newsvendor problem. *Omega (United Kingdom)*, 63:123–133. Cited on page 10.

[Kornacker et al., 2015] Kornacker, M., Behm, A., Bittorf, V., Bobrovytsky, T., Ching, C., Choi, A., Erickson, J., Grund, M., Hecht, D., Jacobs, M., Joshi, I., Kuff, L., Kumar, D., Leblang, A., Li, N., Pandis, I., Robinson, H., Rorke, D., Rus, S., Russell, J., Tsirogiannis, D., Wanderman-Milne, S., and Yoder, M. (2015). Impala: A modern, open-source SQL engine for Hadoop. *CIDR 2015 - 7th Biennial Conference on Innovative Data Systems Research*. Cited on page 33.

[Kotthoff et al., 2017] Kotthoff, L., Thornton, C., Hoos, H. H., Hutter, F., and Leyton-Brown, K. (2017). Auto-WEKA 2.0: Automatic Model Selection and Hyperparameter Optimization in WEKA. *J. Mach. Learn. Res.*, 18(1):826–830. Cited on page 25.

[Kumar and Patel, 2010] Kumar, M. and Patel, N. R. (2010). Using clustering to improve sales forecasts in retail merchandising. *Annals of Operations Research*. Cited on pages 15 and 44.

[Liu et al., 2016] Liu, C.-T., Milton, J., and McIntosh, A. (2016). Simple Linear Regression. Cited on page 22.

[MacDonald, 2003] MacDonald, J. M. (2003). Demand, Information, and Competition: Why Do Food Prices Fall at Seasonal Demand Peaks? *The Journal of Industrial Economics*, 48(1):27–45. Cited on page 9.

[Mantrala and Rao, 2001] Mantrala, M. K. and Rao, S. (2001). A Decision-Support System that Helps Retailers Decide Order Quantities and Markdowns for Fashion Goods. *Interfaces*, 31(3 - Supplement):146–165. Cited on page 16.

[Marshall, 1898] Marshall, A. (1898). *Elements of Economics*. London: Macmillan. Cited on pages 10 and 55.

[Mentzer and Moon, 2005] Mentzer, J. T. and Moon, M. A. (2005). *Sales forecasting management: A demand management approach*. Sage Publications, Inc. Cited on page 8.

[Merrick, 2001] Merrick, A. (2001). Retailers Attempt to Get a Leg Up On Markdowns With New Software. https://www.wsj.com/articles/SB997129731937029295. Accessed on 2020-02-07. Cited on page 15.

[Nachtmann et al., 2010] Nachtmann, H., Waller, M. A., and Rieske, D. W. (2010). The Impact of Point-of-Sale Data Inaccuracy and Inventory Record Data Errors. *Journal of Business Logistics*. Cited on page 36.

[Neelakantan et al., 2007] Neelakantan, J., Gosavi, A., Batta, R., and Talukdar, D. (2007). Optimal clearance-markdown pricing of retail goods using demand-response functions under variance-sensitive and risk-neutral criteria. *Technical Report SOPTL*. Cited on page 16.

[Niemeier et al., 2013] Niemeier, S., Zocchi, A., and Catena, M. (2013). Reshaping retail : why technology is transforming the industry and how to win in the new consumer driven world. Cited on pages 5 and 6.

[Olmstead et al., 2007] Olmstead, S. M., Michael Hanemann, W., and Stavins, R. N. (2007). Water demand under alternative price structures. *Journal of Environmental Economics and Management*. Cited on page 13.

[Omnia Retail, 2020] Omnia Retail (2020). The Ultimate Guide to Dynamic Pricing. https://www.omniaretail.com/the-ultimate-guide-to-dynamic-pricing. Accessed on 2020-02-04. Cited on page 15.

[Pachon et al., 2006] Pachon, J., Iakovou, E., and Chi, I. (2006). Vehicle fleet planning in the car rental industry. *Journal of Revenue and Pricing Management*. Cited on page 13.

[Pafka, 2019] Pafka, S. (2019). benchm-ml. https://github.com/szilard/benchm-ml. Accessed on 2020-05-20. Cited on page 34.

[Palmer, 2004] Palmer, M. (2004). International Retail Restructuring and Divestment: The Experience of Tesco. *Journal of Marketing Management*, 20:1075–1105. Cited on page 6.

[Parkin et al., 2014] Parkin, M., Powell, M., and Matthews, K. (2014). *Economics: European Edition*. Always learning. Pearson Education Limited. Cited on pages 11 and 12.

[Png, 2002] Png, I. (2002). *Managerial Economics*. Blackwell Publishers. Cited on page 13.

[Polli and Cook, 1969] Polli, R. and Cook, V. (1969). Validity of the Product Life Cycle. *The Journal of Business*. Cited on page 1.

[Qamar-ud Din, 2019] Qamar-ud Din, M. (2019). Cross-Validation strategies for Time Series forecasting. Cited on page 54.

[Rousseeuw, 1987] Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20(C):53–65. Cited on page 46.

[Saberidokht, 2019] Saberidokht, B. (2019). Comparing Mature, General-Purpose Machine Learning Libraries. Cited on page 34.

[Seaman, 2018] Seaman, B. (2018). Considerations of a retail forecasting practitioner. *International Journal of Forecasting*. Cited on page 9.

[Senter, A., 2008] Senter, A. (2008). Time Series Analysis. http://userwww.sfsu.edu/efc/classes/biol710/timeseries/timeseries1.htm. Accessed on 2020-05-17. Cited on page 7.

[Smith, 2015] Smith, S. A. (2015). Clearance pricing in retail chains. In Agrawal, N. and Smith, S., editors, *International Series in Operations Research and Management Science*, chapter pp 387-408. Springer, Boston, MA. Cited on pages 16, 17, 18, 49, and 53.

[Smith and Achabal, 1998] Smith, S. A. and Achabal, D. D. (1998). Clearance pricing and inventory policies for retail chains. *Management Science*. Cited on pages 1, 14, 16, 41, and 56.

[Smith and Agrawal, 2017] Smith, S. A. and Agrawal, N. (2017). Optimal markdown pricing and inventory allocation for retail chains with inventory dependent demand. *Manufacturing and Service Operations Management*. Cited on pages 16 and 18.

[Steenburgh and Avery, 2010] Steenburgh, T. and Avery, J. (2010). Marketing Analysis Toolkit: Pricing and Profitability Analysis. *Harvard Business Review*, pages 1–8. Cited on page 11.

[Stellwagen, 2013] Stellwagen, E. (2013). Forecasting 101: Understanding Pareto (ABC) Analysis. Cited on page 42.

[Swamidass, 2000] Swamidass, P. M., editor (2000). *Bias, in forecasting*, pages 61–62. Springer US, Boston, MA. Cited on page 66.

[Thimmapuram and Kim, 2013] Thimmapuram, P. R. and Kim, J. (2013). Consumers' price elasticity of demand modeling with economic effects on electricity markets using an agent-based model. *IEEE Transactions on Smart Grid*. Cited on page 13.

[Thomassey, 2014] Thomassey, S. (2014). Sales forecasting in apparel and fashion industry: A review. In *Intelligent Fashion Forecasting Systems: Models and Applications*. Springer. Cited on pages 1, 9, 10, and 41.

[Vakhutinsky et al., 2012] Vakhutinsky, A., Kushkuley, A., and Gupte, M. (2012). Markdown optimization with an inventory-depletion effect. *Journal of Revenue and Pricing Management*. Cited on page 41.

[Vauhkonen, 2018] Vauhkonen, J. (2018). *Cross-Price Elasticities in Fashion Retail Pricing*. PhD thesis, Aalto University School of Science. Cited on page 17.

[Wirth, 2000] Wirth, R. (2000). CRISP-DM : Towards a Standard Process Model for Data Mining. In *Proceedings of the Fourth International Conference on the Practical Application of Knowledge Discovery and Data Mining*, pages 29–39. Cited on page 20.

[Wolpert and Macready, 1997] Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*. Cited on page 42.

[Wu, 1997] Wu, C. J. (1997). Statistics=Data Science? Cited on page 19.

[Zaarour et al., 2016] Zaarour, N., Melachrinoudis, E., and Solomon, M. M. (2016). Maximizing revenue of end of life items in retail stores. *European Journal of Operational Research*, 255(1):133–141. Cited on pages 13, 16, 17, 18, 19, 49, and 57.

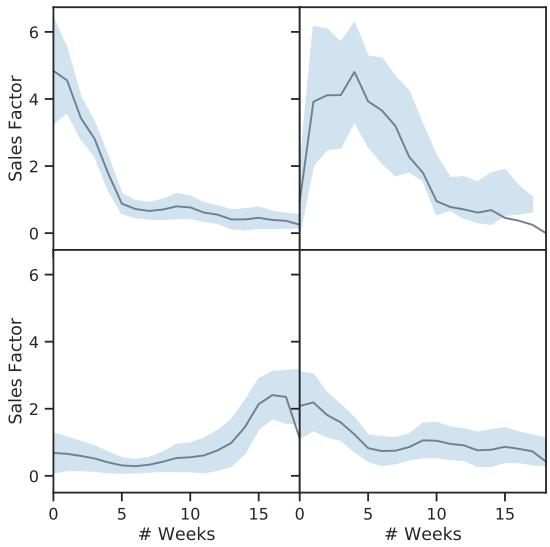
[Zakaria, 2018] Zakaria, Z. (2018). Macroeconomic Environments and the Demand for Retail Space in the Shopping Centres in Malaysia. In *International Conference on Innovation & Management (ICIM2018)*. Cited on page 9.

[Zhao and Zheng, 2000] Zhao, W. and Zheng, Y. S. (2000). Optimal dynamic pricing for perishable assets with nonhomogeneous demand. *Management Science*. Cited on pages 15 and 17.

Appendix A

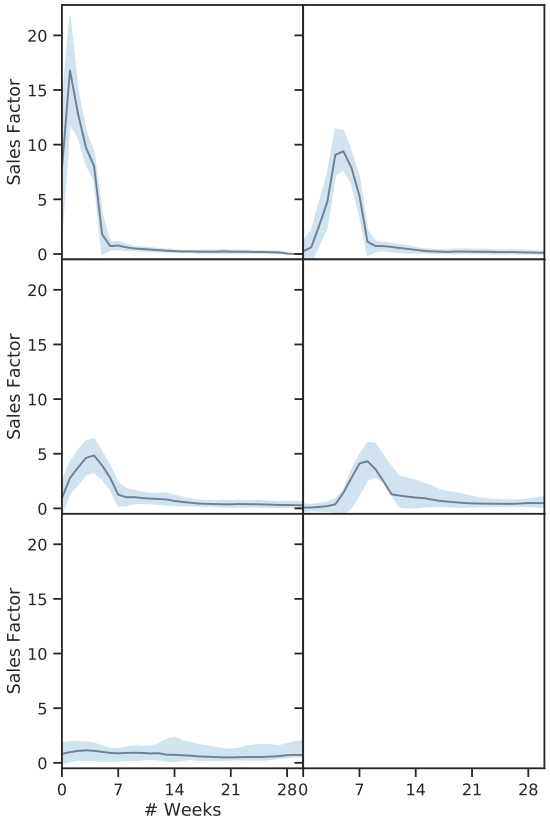
Clusters

Resulting clusters for each *Season Group* of the target categories (dark line). The shaded areas illustrate the SKUs that belong to the clusters, represented by the standard deviation of the correspondent *sales factor* values.

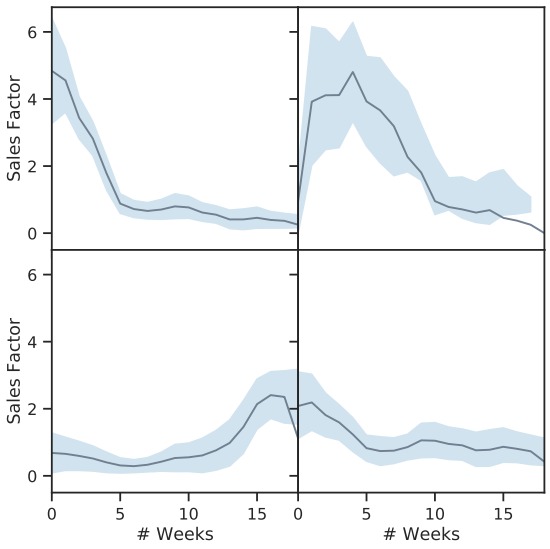


(a) Trimesters.

Figure A.1: Resulting clusters of the *Office* category.



(a) "Home Special"



(b) Fall

Figure A.2: Resulting clusters of the *Home* category.